AFRL-HE-WP-TR-2002-0232

UNITED STATES AIR FORCE
RESEARCH LABORATORY

# Developing A Multi-Tasking Cognitive Agent Using the COGNET/iGEN Integrative Architecture

Wayne Zachary
Thomas Santarelli
Joan Ryder
James Stokes

CHI Systems, Inc.
716 N. Bethlehem Pike
Suite 300
Lower Gwynedd, PA 19002

December 2000

Final Report for the Period October 1998 to December 2000

Human Effectiveness Directorate
Deployment and Sustainment Division
Sustainment Logistics Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604

| REPORT DOCUMENTATION PAGE | *Form Approved* *OMB No. 0704-0188* |
|---|---|

| | | |
|---|---|---|
| **1. AGENCY USE ONLY** *(Leave blank)* | **2. REPORT DATE** December 2000 | **3. REPORT TYPE AND DATES COVERED** Final - October 1998 - December 2000 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** Developing A Multi-Tasking Cognitive Agent Using the COGNET/iGEN Integrative Architecture | **5. FUNDING NUMBERS** C: F33615-99-C-6007 PE: 63106F PR: 2745 TA: 04 WU: 04 |
| **6. AUTHOR(S)** Wayne Zachary, Thomas Santarelli, Joan Ryder, James Stokes | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** CHI Systems, Inc 716 N. Bethlehem Pike Suite 300 Lower Gwynedd, PA 19002 | **8. PERFORMING ORGANIZATION REPORT NUMBER** 001004.9915 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** Air Force Research Laboratory, Human Effectiveness Directorate Deployment and Sustainment Division Air Force Materiel Command Sustainment Logistics Branch Wright-Patterson AFB OH 45433-7604 | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AFRL-HE-WP-TR-2002-0232 |
| **11. SUPPLEMENTARY NOTES** | |
| **12a. DISTRIBUTION AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited. | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT** *(Maximum 200 words)*

This report describes CHI Systems' model development effort as part of the initial model comparison process of the Agent-Based Modeling and Behavioral Representation (AMBR) program being conducted by the Air Force Research Laboratory. CHI Systems developed its model of human multi-tasking behavior using the COGNET integrative executable cognitive architecture and iGEN™ model development environment. The example problem environment in which all AMBR models in this comparison cycle were executed is a simplified Air Traffic Control (ATC) environment, developed by an independent moderator contractor, BBN Technologies. The moderator contractor also collected human performance data from live subjects as a point of comparison for model performance.

| | | | |
|---|---|---|---|
| **14. SUBJECT TERMS** Human Performance Models Human Behavior Representation Joint Synthetic Battlespace Simulation | | | **15. NUMBER OF PAGES** 68 **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** UNCLASSIFIED | **18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED | **19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED | **20. LIMITATION OF ABSTRACT** UL |

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

THIS PAGE INTENTIONALLY LEFT BLANK

## PREFACE

This research was accomplished as part of the Agent-Based Modeling and Behavioral Representation (AMBR) project, managed by the Air Force Research Laboratory, Human Effectiveness Directorate, Sustainment Logistics Branch, contract number F33615-99-C-6007. The period of performance for this task was from October 1998 through December 2000. Dr. Kevin A. Gluck, AFRL/HEA, Williams AFB, Mesa, AZ, was the technical monitor for this effort.

**TABLE OF CONTENTS**

1. Introduction ........ 1
1.1 Theories and Models ........ 1
1.2 Overview of Approach and Report ........ 2
2. COGNET/iGEN™ Framework ........ 3
2.1 Information Processing Mechanisms in CGF-COGNET ........ 4
2.1.1 Background and Foundation ........ 5
2.1.2 CGF-COGNET Principles of Information Processing Operation ........ 7
2.1.3 CGF-COGNET Information Processing Mechanisms and Architecture... 13
2.2 COGNET Expertise Framework ........ 17
2.2.1 Declarative Expertise ........ 20
2.2.2 Procedural Expertise ........ 21
2.2.3 Action Expertise ........ 22
2.2.4 Perceptual Expertise ........ 22
2.2.5 Metacognitive Expertise ........ 25
2.2.6 Interactions with the External World in COGNET ........ 27
2.3 Three Types of COGNET Models ........ 28
2.4 Software Support for the Model-building Process ........ 28
3. Model Description ........ 33
3.1 Modeling approach ........ 33
3.2 Model characteristics ........ 34
3.2.1 Model structure and conceptual overview ........ 34
3.2.2 Multi-tasking strategy and attention management ........ 36
3.2.3 Visual scanning ........ 37
3.2.4 Blackboard ........ 38
3.2.5 Cognitive tasks ........ 39
3.2.6 Actions and action time charges ........ 42
3.2.7 Errors ........ 43
3.3 Model development process ........ 44
3.4 Subjective workload assessment ........ 46
3.4.1 Relating TLX Measures to COGNET Constructs ........ 47

3.4.2 Operationalization of the Measures in BATON Terms ........................... 49
3.4.3 Implementation of Measures in BATON ................................................ 50
3.4.4 Calibrating the Measures to the TLX Scale ............................................ 52
3.4.5 Workload Assessment Results ................................................................ 53
4. Model Results and Evaluation ..................................................................... 53
4.1 Model results ............................................................................................ 53
4.2 Development process evaluation .............................................................. 57
4.3 Competition process evaluation ................................................................ 58
5. Conclusions .................................................................................................. 58
6. References .................................................................................................... 59

## LIST OF FIGURES

Figure 2-1. COGNET Processing Mechanisms Roots ............................................. 8
Figure 2-2. COGNET Information Processing Architecture ................................... 13
Figure 2-3. CGF-COGNET Information Processing Architecture .......................... 14
Figure 2-4. Influences in COGNET Expertise Representation ................................ 19
Figure 2-5 Blackboard Representation of Declarative Information in COGNET .......... 20
Figure 2-6. Three Types of COGNET Models ........................................................ 29
Figure 2-7. Organization of the iGEN™ Cognitive Agent Toolkit ............................ 30
Figure 2-8. Use of iGEN™ to Develop the AMBR ATC Model. ................................ 31
Figure 2-9. Execution of COGNET/iGEN™ Model ................................................ 32
Figure 3-1. AMBR Model Conceptual Overview .................................................... 35
Figure 3-2. Task Trigger Using Detect Event .......................................................... 37
Figure 3-3. ATC-Model Standard Blackboard Structure .......................................... 39
Figure 3-4. ATC-Model Metacognitive Blackboard Structure .................................. 39
Table 3-1. ATC Model Task Listing ........................................................................ 40
Figure 3-5. High-level View of SA Task ................................................................. 41
Figure 3-6. Update Situation Awareness Task Extract ............................................. 41
Figure 3-7. Interaction Task Example — Accept Incoming Aircraft ........................ 42
Figure 3-8. Comparison of Update Situation Awareness Task for Text and Color ......... 45
Figure 3-9. Model versus Human Subjects on Mental Demand Measure ................... 54
Figure 3-10. Model versus Human Subjects on Physical Demand Measure ............... 54

Figure 3-11. Model versus Human Subjects on Temporal Demand Measure ................ 55
Figure 3-12. Model versus Human Subjects on Performance Measure ................ 55
Figure 3-13. Model versus Human Subjects on Effort Measure ................ 56
Figure 3-14. Model versus Human Subjects on Frustration Measures ................ 56

## LIST OF TABLES

Table 2-1. COGNET Principles of Operation ................ 9
Table 2-2. COGNET Description Language for Procedural Expertise ................ 23
Table 2-3. Organization of the Default Metacognitive Blackboard Panel ................ 26
Table 3-1. ATC Model Task Listing ................ 40
Table 3-2. ATC Model Action Time Charges in Micro Models ................ 43
Table 3-3. Error Types in ATC Task Performance ................ 44
Table 3-4. BATON Operationalization of AMBR Workload Measures ................ 50
Table 3-5. Calibration formula for AMBR Workload Measures ................ 52

# 1. INTRODUCTION

This report describes CHI Systems' model development effort as part of the initial model comparison process of the Agent-based Modeling and Behavioral Representation (AMBR) program being conducted by the Air Force Research Laboratory. CHI Systems developed its model of human multi-tasking behavior using the COGNET integrative executable cognitive architecture and iGEN™ model development environment. The example problem environment in which all AMBR models in this comparison cycle were executed is a simplified Air Traffic Control (ATC) environment, developed by an independent moderator contractor, BBN Systems and Technologies, Inc. The task and simulator are described in separate reports by BBN. The moderator contractor also collected human performance data from live subjects as a point of comparison for model performance. The data collection and results are also discussed in a separate report by BBN.

## 1.1 Theories and Models

Before discussing either COGNET or the ATC model in more detail, it is useful to situate COGNET in the broader framework of cognitive modeling/simulation research, so that its goals and origins can be more clearly understood. We begin by contrasting the notion of theory and model. 'Theory' can be defined as a body of general principles and/or abstractions, formal or otherwise, about the world (both phenomenal and epistemic). Theory formation, then, is a process of creating such abstractions and principles, without any inherent linkage to any specific purpose. It is simply the construction of a narrative on the universe. In contrast, a 'model' can be defined as a specific representation of some set of phenomena[1] that attempts to recreate or predict some aspect of those phenomena *for a specific purpose*. Model-building is thus viewed as an inherently purposive process, one undertaken to support a specific end or to answer a specific question, such as 'how should this device be engineered?' or 'how might this building aesthetically change the neighborhood?' or even 'how predictive or parsimonious is a specific psychological theory?' Thus, representing a city as a point in a directed graph model may be completely adequate for the purpose of calculating the optimal travel time of a multi-city trip (as in the so-called 'traveling salesman' problem), but completely inadequate for the purposes of actually traveling to that city (for which a different type of model – the road map – would be better), and vice versa.

When viewed in this way, models are all simplifying representations of the phenomena they address. Why? If they did not somehow simplify the phenomena, the purposes might just as well be addressed by experimentation – interaction with the phenomena directly. This leads to the proposition that all models are *as-if* models, filled with simplifying assumptions (which in the best case are all explicit and deliberate). That is, they attempt to predict or recreate the first order phenomenon *as if* it were equivalent to the simplified representation inherent in the model.

[1] The term phenomena is somewhat misused in the following discussions, in that it is intended to include physically sense-able information (the usual denotation) but also epistemic information such as knowledge and feelings which rely on consciousness rather than sensation. The reasons for this should be obvious. The model being presented here is both phenomenological (predicting behavior) and epistemological (predicting knowledge use).

Divergences between the model and the denotata of the model help identify the benefits or limitations of such simplifications, given the purpose of the model. In this way, model-building can be used as a way of wielding Occam's Razor, by providing a link between the level of parsimony of a representation and its usefulness.

On the other hand, the fact that models are inherently simplifying, as-if representations, means that the phenomena they represent are always more complex than the models themselves. When a model is used to embody a theory, there is a tension between modeling in the purposive sense (which strives for parsimony) and theory development (which strives for completeness). This can create a kind of 'slippery slope', where additional detail is added to the model (or better put, simplifications are removed) because the process creates a more direct correspondence with the phenomena being modeled. The result, however, can be a representation that is called a model but which has no specific purpose, and which may not stand up to Occam's Razor when it is used for a specific purpose. For example, a model of the unique cellular physiology of the heart muscle may be more complete in many senses than a model which represents the heart as pair of interconnected pumps, yet for the purpose of predicting the rate and quantity of flow of blood to the lungs, the latter may be a much more effective and parsimonious representation.

This sets the stage for a brief description of the COGNET framework. COGNET was created as a method or framework for building models of human information processing for a range of specific purposes (which has grown over time). These purposes include design/analysis of person-machine interfaces and human work tasks, performance diagnosis in intelligent training systems, decision-support and performance-support in real-time computer-based work environments, and simulation of human behavior and performance, the purpose in this project. These purposes are all quite distinct, and ultimately require different kinds of models of human information processing. Not as different, perhaps, as the map and the connected graph mentioned earlier, but different nonetheless in the level of detail and types of structures, processes, and output information required. As new COGNET-based models have been built, a deliberate effort has been made to identify a structure that allows common features to be identified and retained in the framework, and points of difference to be identified and 'left open' so as to not constrain future model-building efforts. This process has been motivated by theory, ideally in the sense noted above: to identify useful simplifying assumptions and boundary conditions of the known assumptions. COGNET does not attempt, however, to either capture or represent any unified theory of human thought or behavior, though (we believe) it could be used to both express and test such theories at some level. At the same time, the structure of COGNET has been driven by pragmatics and experience, providing a framework in which (purposive) models can be constructed easily and efficiently by individuals not necessarily expert in theories of human cognition.

## 1.2 Overview of Approach and Report

Our approach to creating a model for the AMBR ATC task can be viewed in light of the discussion of theories and models above—it was to create a model that could carry out the ATC-like task posed in a human-like manner at the level of strategy or function. Further detail in lower-level operations would be added as necessary to allow the model to generate the behavior required. The COGNET framework provided the underlying cognitive architecture and Principles of Operation for the model. The COGNET framework and associated iGEN™ development tools are described in Section 2.

We created two models, one for each of the fly-off conditions: text (unaided) and color (aided). These models were actually just variations of each other. Our models represented one generic individual; that is, they produced a generic individual's performance rather than a range of performance over multiple runs. Each model adjusted dynamically to workload within conditions. The modeling approach is described in more detail in Section 3.1. The model itself is discussed in Section 3.2, followed by a short description of the model development process (Section 3.3). At the end of each model run, the model output the six subjective workload measures requested based on 'introspection of its cognitive processes.' The subjective workload assessment measures are described in Section 3.4.

Model results and our evaluation of the model development and competition processes are discussed in Section 4. General conclusions follow in Section 5.

## 2. COGNET/IGEN™ FRAMEWORK

COGNET is a model of human information processing that is based on an explicit decomposition of the phenomena involved. This decomposition is reflected in the metaphorical 'equation' shown in [1], which is analogous to the equation used by Card, Moran, and Newell to decompose human-computer interaction (1983:p. 27). The focus of this equation is *competence*, in the sense it is used by linguists, that is, the ability of a person to construct appropriate behaviors in a specific context, unburdened by various constraining factors of performance or pragmatics.

[1] Individual Competence = processing mechanisms +
internal expertise +
context

In the decomposition given in [1], competent problem-solving emerges from the manipulation of a body of internal expertise by a set of (presumably biological) internal information processing mechanisms, as required by the features of and interactions with the external context of the behavior. The effect of the external situation is critically important in this equation. It provides the details and data from which the work goals are identified and activated, and in pursuit of which expertise is activated, retrieved, and manipulated.

The relationships in [1] indicate how human information processing could be decomposed (and represented) if one wanted to capture and predict, under essentially perfect circumstances, the kinds of problem-solving situations for which a person was competent. The analogy to linguistics can clarify this notion. Linguistic competence refers to the ability of a person to understand and produce completely correct, fully formed meaningful utterances in a speech context. Thus, problem-solving competence refers to the ability of the person to understand a situation and produce appropriate, well-structured, interpretable, and goal-based behaviors as that situation unfolds. But just as most people rarely speak in completely correct, fully-formed sentences, so it is that individual problem-solvers seldom produce fully correct and optimal behaviors in any real world context. Even when the person has complete competence, in the sense of complete expertise, fully functioning processing mechanisms, and full access to the environment, other factors can cause actual behavior, which is termed *performance*, to deviate from pure competence. These include factors of timing – the fact that processing mechanisms can require non-zero amounts of time to accomplish various functions – and accuracy – the fact that these mechanisms can function in a way that may sometimes deviate from the ideal. These effects are seen in COGNET as further constraints upon the model of problem solving competence, as shown in [2].

[2] Individual Performance = processing mechanisms +
internal expertise +
external context +
time + accuracy

The decomposition in [2] provided the starting point for the AMBR project, and led to the use of a variant of the COGNET framework and associated software based on that decomposition. This variant is currently being developed under separate sponsorship, with the goal of providing a better means to create and simulate computer-generated forces or CGFs. Appropriately enough, this variant is termed CGF-COGNET, and the discussion that follows applies specifically to it. However, the 'core' COGNET and the CGF-COGNET variant are highly similar in their representation of both internal processing mechanisms and internal expertise. The main differences are that the CGF-COGNET model and software contain added structure that supports, either directly or infrastructurally, the representation of timing and accuracy effects.

The decomposition in [2] provides a structural framework for the CGF-COGNET system. The competence level consists of representations of the internal processing mechanisms, and of the internal expertise. The external context is defined on a domain-specific basis. The performance level adds additional constraining and limiting factors to these components, particularly to the internal components – the processing mechanisms and the expertise. This framework is used below to describe the CGF-COGNET system. The processing mechanisms are discussed in 2.1, the internal expertise is discussed in 2.2, as is the representation of and connection to the external environment. The various types of models that can be developed using the COGNET framework are discussed in 2.3.

All variants of the COGNET system (including CGF-COGNET) are embodied in a software environment called iGEN™. This software environment provides a number of tools, which allow domain-specific COGNET models to be built and executed, and interfaces which allow a domain-specific model to dynamically interact with a (real or simulated) external environment. iGEN™ and its relation to COGNET are discussed in 2.4 below.

## 2.1 Information Processing Mechanisms in CGF-COGNET

The CGF-COGNET information processing mechanisms can be defined in terms of their structure, (i.e., what the mechanisms are and how they are interconnected), and their function (i.e., what they do and the principles by which they operate). These are discussed below, following a brief overview of the theoretical and intellectual foundations of the COGNET framework.

## 2.1.1 Background and Foundation

The distinction between processing mechanisms and the information being processed has a long history in computer science, going back at least to Turing (1950) and his notion of an abstract computational device. In the last half century, the ability of physical computational devices to create behavior that historically was considered to require human intelligence has given rise to a computer metaphor of mind. Using the terms of the introduction, this metaphor is really an as-if model, in which the cognitive phenomena are approached and analyzed as if they worked like some complex computational device, the structure and principles of which have yet to be discovered. Most researchers following this avenue of approach have certainly not asserted that the mind was a computer, but that it could be productively studied as if it were (many sources, for example Hofstadter, 1979, Pylyshyn, 1985, and Newell, 1990, have elaborated on this concept in varying ways).

The COGNET framework definitely arises from within this tradition, and thus separates its representation of postulated internal information mechanisms from its representation of the information on which those mechanisms operate. However, it is presumed that these two have a 'hand in glove' relationship, in which the information is organized and structured in a way that optimizes, or at least facilitates, its processing by the underlying mechanisms.[2]

The overall architecture of the COGNET processing mechanism follows a well-established cognitivist breakdown along the lines established by Broadbent (1958), Card, Moran and Newell (1983), and Newell (1990), among others. It postulates fully parallel perceptual, motor and cognitive sub-systems, with the cognitive and perceptual sub-systems sharing independent access to a memory structure. The memory construct in the underlying COGNET framework is a long-term working memory structure, which subsumes operations that have been ascribed in the literature to short term, long term and working memory. Ericcson and Kintsch (1995) argue for this type of organization from empirical data, while McGaugh (2000) supports similar arguments from an essentially neurophysiological perspective. Here again, the as-if nature of the COGNET model needs to be noted. COGNET does not presume that short-term and long-term memory differences do not exist, but merely that cognitive processes can be modeled without these distinctions. Ideally, analysis of the resulting models can shed light on when and where such constructs are needed to achieve specific modeling goals.

The Principles of Operation of these architectural components have been derived from a number of more detailed theories and models, largely based on the purposes for which the framework was designed. As noted in Section 1, models (unlike theories) are constructed for specific purposes, and COGNET was developed for application-oriented purposes. Specifically, it was developed to support the *description, prediction, and incorporation of expert-level*

[2] The reasoning for this is evolutionary in nature, and briefly summarized as follows. The mechanisms, as biologically based, are subject to evolutionary processes at the population level. Internal information, however, is acquired during the individual's lifespan, and individuals learn to represent and organize knowledge in a way that is easy (and effective and efficient to process) across their lifespan. Thus, expert-level knowledge is likely to reflect the structures of the underlying processing mechanisms, at least to the degree that the individual is aware of them. Normally, individual level knowledge is lost upon death, removing it from the scope of biological evolutionary mechanisms. Human culture, however, mediates this process by providing a repository of structures (and content) that have some survival value across the lifespan of individuals and are exposed to a more diffuse cultural evolutionary process.

*competence in complex real-time, multi-tasking, work environments.* These work environments are frequently encountered in manufacturing, medical, aerospace, and telecommunications contexts (among others), where the value of improved processes and tools is high. In other words, COGNET was created to address difficult problems in high-payoff application environments. This overall purpose led to several specific characteristics of the system, as discussed below.

Time granularity – the nature of the complex work environments which COGNET seeks to address unfold over periods of time which range from second to hours. This places them in what Newell (1990) called the 'rational band' and the upper portions of the 'cognitive' band of temporal granularity (ibid., Figure 3-3). COGNET focuses on this range of granularity, eschewing constructs which operate at coarser and (particularly) finer levels of granularity, assuming that they are either too large or too small to have a direct influence on processes within the focused range. Here again, this is an as-if assumption, allowing the phenomena within the range of interest to be modeled as if they were independent of the lower and higher level processes and structures. It is likely, in fact, that near certain boundary conditions this assumption may not hold, but identification of these boundary regions is also a useful activity.

Attention – the focus on representation and prediction in complex multi-tasking environments forces COGNET to deal explicitly with competing demands for action and competing opportunities to gain information, as well as constraints from the external environment on what can be done (time sequencing in external environment). Thus attention and the forces, which shape it, have been central concerns within COGNET from the beginning. There are two main concepts behind the COGNET approach to attention. The first is the notion of weak concurrence, which assumes that a person can actively pursue only one (high-level) goal at a time, although there can be many threads of goal-pursuing activity that are active at any one time. These threads may represent interrupted lines of reasoning, temporarily suspended ones, or even goals that the person knows are relevant but have not yet been activated. The fact that multiple lines of reasoning are being pursued simultaneously makes the process concurrent, but the fact that only one of these is being actively pursued makes it weakly so. The second concept underlying attention in COGNET is the notion that attention emerges from primarily cognitive processes, rather than existing as a separate executive process itself. The basis for this is the lengthy discussion in the literature during the 1960s through the 1970s, in which it was eventually realized that an explicit attention process required some higher level executive process to manage it, and that this regress could continue ad infinitum. The fundamental concept of attention, which COGNET has incorporated and greatly elaborated, is the Pandemonium model first proposed by Selfridge (1959), which provides a representation of attention that is both weakly concurrent and emergent.

Embodiment – The need to represent the human role in complex environments has required COGNET to consider explicitly the physical mechanisms which link perceptual/action processes to the external environment. These physical mechanisms force COGNET to be an embodied cognition system (Gray & Boehm-Davis, in press), in which the interaction with the external environment affects internal processes in a fundamental and on-going way. In CGF-COGNET these interactions have become even more important drivers of the overall system performance. However, the time granularity of the overall system (seconds to hours) is to some degree inconsistent with the time granularity at which many of the effects of embodiment occur (microseconds to seconds). Thus, COGNET has adopted a granularity-neutral stance with regard to embodiment, allowing the modeler to incorporate constraints and features of the physical systems to the degree necessary and appropriate for the specific application. This is in contrast to systems such as EPIC, for example, which adopt a fixed (and relatively fine) granularity for the structure and processes of the embodiments of the system. The CGF-COGNET variant nonetheless seeks to more explicitly capture the constraining effects of embodiment. To permit this in a granularity flexible manner (in which there would be no fixed models of body features), the performance prediction approach of *micromodels* was adopted. This approach, which originated with the Human Operator Simulator (HOS) system of the 1970s (Lane, Strieb, Glenn, and Wherry, 1981), uses closed form approximations of the time and/or accuracy constraints of specific physical instrumentalities in specific types of contexts (e.g., accuracy of reading characters of text; time to fingertip-touch an object within the current reach envelope, etc.). These micromodels allow existing experimental data and empirical relationships to be encapsulated and reused, but do not force any specific level of representation of body features in CGF-COGNET.

Expert-oriented knowledge structure – The COGNET focus on expert-level competence and performance in complex environments led to the representation of internal information in a manner as similar as possible to that used by human experts. In turn, this led to the incorporation of theories of expertise and expert knowledge structures that emphasized the efficiency and parsimony of expert decision processes, particularly in the real-time contexts where COGNET is focused. These theories (see Chi, Glaser and Farr, 1988; Ericcson and Smith, 1991; Hoffman, 1992), suggest that experts use highly compiled knowledge structures that minimize the process of searching complex knowledge spaces. These theories meshed with the notions of recognition primed decision-making, first suggested by Klein (1989), and the artificial intelligence concept of case-based reasoning (Kolodner, 1988). Although deriving from very different bodies of data, both of these suggested that context cues based on internal models of the external situation, allowed these compiled knowledge structures to be activated on a just-in-time basis in real-time settings. COGNET ultimately chose to incorporate all these concepts in its framework for representing the internal information or expertise held by the expert.

The various influences that shaped the COGNET representation of information processing mechanisms are summarized in Figure 2-1. The COGNET Principles of Operation derive from these general influences, as do the details of the internal mechanisms and architecture of COGNET and CGF-COGNET. Both are reviewed in the subsections that follow. While the Principles of Operation apply equally to COGNET and the CGF-COGNET variant, the information processing mechanisms differ between the two; the CGF-COGNET variant is primarily discussed below.

### 2.1.2 CGF-COGNET Principles of Information Processing Operation

The structure and processing of information in the COGNET system is based on a series of Principles of Operation, which are listed in Table 2-1 below. Each principle is discussed in more detail in the following subsections.

#### *2.1.2.1 Attention Focus Principle*

The attention focus principle simply states one aspect of the concept of weak concurrence, by specifying that only one unit of procedural (goal-directed) knowledge can be active at a time. It also defines two properties of this unit of cognitive process execution:

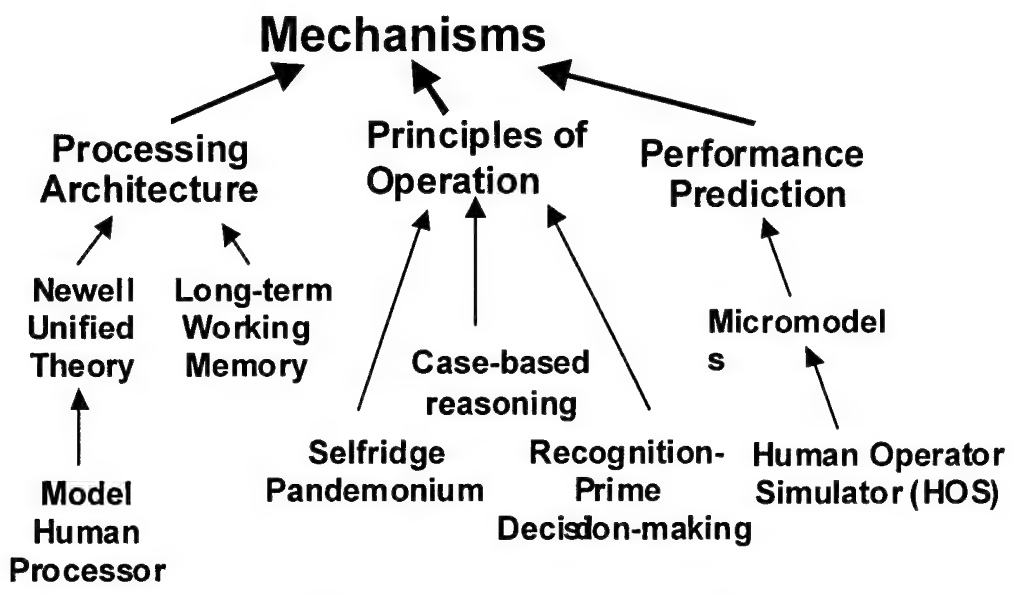


**Figure 2-1. COGNET Information Processing Mechanisms Roots**

- that it represents a chunk (rather than an atomic unit) of procedural knowledge, and that the unit is called a cognitive task, and
- that it can be in (at least) two different states – executing, and non-executing (the latter of which actually encompasses several different states, for reasons that will become clear later)

Thus, this principle begins (and others will follow) the process of defining the form of the internal information that is processed by the system. In particular, the notion that procedural knowledge is chunked into large units and executed in these units is consistent with the underlying models of expert decision processes discussed above.

#### *2.1.2.2 Pattern-Based Attention Demand Principle*

This next principle, as well as the following two, provides more definition for the way attention operates within COGNET and how knowledge is structured to fit within this process. To begin with, it defines a relationship between declarative information and procedural information. Specifically, it states that some combination or pattern of information in memory, simply by virtue of its existence, can result in a procedural chunk (i.e., cognitive task) changing its state from inactive to a new state that is termed active. The pattern or condition that causes this activation is incorporated within the cognitive task itself, and is termed the trigger. In a real sense, then, the trigger is actually a piece of metacognitive knowledge that 'wraps' the procedural chunk. Although it does not say it explicitly, the pattern-based attention demand principle implies that the process of comparing the trigger to the contents of memory is something that is done within the processing mechanism itself, as part of the cognitive process. In addition, this activation of large procedural knowledge chunks on the basis of broad patterns or contexts are a realization of the concepts of recognition primed decision-making and case-based reasoning discussed earlier.

**Table 2-1. COGNET Principles of Operation**

| Principle name | Definition |
|---|---|
| Attention Focus | *At any point, the cognitive process is executing, at most, one chunk of procedural knowledge, called a cognitive task. The executing cognitive task is said to have the focus of attention of the cognitive process.* |
| Pattern-based Attention Demand | *Each cognitive task includes a trigger condition to determine if it is relevant to the internal understanding of the external world, as maintained in memory. Only when the trigger condition is met does the task become active and vie for attention.* |
| Attention Capture | *Each cognitive task includes a measure of its relative priority, which is evaluated based on the contents of memory when the task is activated. An activated cognitive task will capture the focus of attention if its priority exceeds those of all other active cognitive tasks.* |
| Task Interruption | *An executing cognitive task that loses the focus of attention through Attention Capture has been interrupted and continues to compete for attention.* |
| Cognitive Process Memory Modification | *Cognitive operators within cognitive tasks can change the state of memory when they are executed.* |
| Perceptual Process Memory Modification | *Self-activating perceptual knowledge sources called demons change the state of memory when executed by the perceptual process.* |
| Multiple Cognitive Task Instances | *A cognitive task may be activated for execution in the context of a specific piece or configuration of knowledge in memory, which is called its scope. Such a cognitive task may have multiple active instances, with each instance reflecting a unique instance scope. Multiple cognitive task instances compete for attention as if they were individual cognitive tasks.* |
| Task Suspension | *A task may relinquish attention, based on task knowledge and the state of memory. Upon such suspension, all remaining active tasks vie for attention. The suspended task sets a resumption condition that determines when it will be reactivated and resume competing for attention.* |

In its active state, the principle further specifies, the cognitive task is vying for the focus of attention. This, in turn, implies that such an activated task does not have the focus of attention, yet it also is not inactive. Thus, this defines a third state of a cognitive task, besides executing and inactive, termed 'active'. The process of attention can now be defined as the process by which tasks move from inactive to active to executing (and ultimately back again). The current principle only deals with the first half of this cycle, showing how a cognitive task moves from an inactive state to an active state (i.e., by its trigger being satisfied by the contents of memory).

### *2.1.2.3 Attention Capture Principle*

This principle defines the other half of the attention process, that is, the mechanisms by which the focus of attention is allocated (and execution begins). The Attention Capture Principle clarifies this by introducing another metacognitive construct, the notion of a momentary priority of an activated task. Like the trigger, the priority is based on the information in memory at the current time. This suggests that the priority will vary as the contents of memory vary, an observation that applies to the trigger as well. Thus as memory changes (later principles will specify how that happens), a trigger may become 'unsatisfied' and the task inactive once again. Similarly, the priority may vary up and down as the contents of memory changes. When combined, the trigger and priority behave like the 'shrieking demons' in Selfridge's Pandemonium model. As with the previous principle, the Attention Capture Principle suggests but does not explicitly state that the process of evaluating the priority (constantly) and changing the focus of attention when some cognitive task's priority exceeds that of the executing task is organic to the cognitive process itself. Given that this is the case, however, the result is that attention emerges from the interaction of the changing memory contents and the metacognitive knowledge encoded in the triggers and priorities.

### *2.1.2.4 Task Interruption Principle*

The previous principles leave unanswered the question of what happens to an executing cognitive task when another activated task captures the focus of attention before the first chunk of procedural knowledge has completely executed. This question is answered by the Task Interruption Principle, which adds yet another state that procedural knowledge chunks may assume – interrupted. It states that a task procedural chunk, which has lost the focus of attention, is in an interrupted state in which it continues to compete for attention. It is implied by the Pattern-based Attention Demand principle that this competition will continue only so long as the associated trigger is satisfied by information in memory. Similarly, the Attention Capture principle implies that the priority of the interrupted task may continue to change as the priority measure changes as information in memory changes. Thus, the interrupted task may regain the focus of attention if at some future time its priority exceeds that of the currently executing task. It may also re-gain the focus of attention if the currently executing task completes execution and the interrupted task is the only activated task or the activated task with the highest priority.

The Task Interruption Principle does not define what happens when an interrupted task resumes execution. In the standard COGNET system, the task will resume at the point where it was interrupted. This can, however, create problems in situations where the external world (and the internal representation of it) has changed substantially while the task was interrupted. As discussed in 2.2.2 below, metacognitive mechanisms have been incorporated into CGF-COGNET to support a richer set of means for adapting to recovery from such conditions when interruption and subsequent resumption occur.

### *2.1.2.5 Cognitive Process Memory Modification Principle*

The previous three principles detail the relationship between the dynamics of memory and the attention process in COGNET. Specifically, they show how changes in memory can cause triggers to activate and deactivate procedural knowledge chunks, and how priority measures lead to some of these chunks gaining, and sometimes losing, the focus of attention. They do not discuss, however, the means by which memory can change in COGNET.

The Cognitive Process Memory Modification principle begins by defining a unit of procedural knowledge within the cognitive task. This unit is called a cognitive operator. The principle also states that this lower level unit of procedural knowledge can, when executed by the cognitive processor, modify the contents of memory in some way. The principle implies that there can be more than one of these cognitive operators within a cognitive task, but does not define any additional details of the lower level components of cognitive tasks. In general, the changes in memory can be seen as the result of inferences of various sorts that are defined by the content of the procedural knowledge itself.

Although this principle is very simple, its implications are substantial for the system. On the surface it simply states that a cognitive task can alter memory during its execution. However, given the linkage between memory contents (and more precisely memory dynamics) and attention, this principle indicates that the attention dynamics are driven by the execution of cognitive tasks (although not exclusively so, as shown by the next principle). Thus, a chunk of procedural knowledge may, in the course of its execution, create changes in memory which may in turn lead to other tasks becoming activated and/or lead to changes in priority measures of activate tasks that cause them to capture the focus of attention.

This principle therefore creates openness in the attention process, allowing any procedural chunk to make changes in memory that can allow any other chunk of procedural knowledge to become activated and/or to capture attention. The principle also provides a de facto granularity for the attention process. Because memory can be changed only as result of execution of cognitive operators within cognitive tasks, this means that changes in the focus of attention can only occur after execution of a cognitive operator.

#### *2.1.2.6 Perceptual Process Memory Modification Principle*

The previous principle showed how cognitive processing can result in memory changes, driving the attention process forward in complex ways. Still, this process is in some sense closed, as memory can only be modified (thus far) by internal processes that are finite knowledge sources. Thus, even if there is some stochasticity in their contents, the possible set of changes and dynamics that can occur are bounded. More important, however, is the fact that it leaves memory, and therefore the cognitive subsystem, disconnected from the external world. This problem is resolved by the Perceptual Process Memory Modification Principle, which shows how the perceptual subsystem injects information that has been perceived about the external world.

This principle defines an entirely new type of procedural knowledge, called the 'demon.' This unit of procedural knowledge is executed by the perceptual process rather than by the cognitive processor. (As a result, the unit is sometimes called a 'perceptual demon' rather than simply a demon). A key property of this unit of internal information is that it is self-activating, in response to a specific sensory cue. Thus, there is no attention process within the perceptual subsystem as there is within the cognitive subsystem. Rather, information is sensed and this sensation process (which can be thought of as registering external cues inside the system) leads organically to the activation and execution of perceptual demons that are able to process the information.

The principle also specifies what these units of procedural knowledge do when they are executed – they modify the contents of memory. It does not indicate whether there are any limitations to how many demons can be activated and executed within any perceptual processor cycle, nor whether there are any limitations to how many modifications to memory can be made in any time or cycle interval. In other words, there is no inherent bandwidth limitation to the perceptual process in COGNET. However, human sensory and perceptual limitations do create bandwidth constraints, and the CGF-COGNET variant therefore does provide some facility to represent these limitations (see 2.1.3.1 and 2.2.4 below).

This principle complements the previous principle in showing a second way in which memory can be modified, i.e., as a result of information sensed and perceived from the external world. This adds much more complexity and openness to the dynamics of the information process, allowing flows in the focus of attention that can be driven either from reasoning processes (i.e., from memory changes resulting from cognitive operators) or from environmental perceptions (i.e., from perceptual demons). The latter type, in particular, allows for very abrupt changes in the focus of attention. For example, unexpected environmental information, such as hearing an alarm, can lead to abrupt shifts in attention to very different cognitive tasks.

This principle does not address any differences in time granularity between the perceptual and cognitive processes. While the previous principle implicitly set the granularity of cognitively-driven attention flows at the level of the execution of individual cognitive operators, the present principle does not indicate whether the memory changes resulting from perceptual process modifications occur at the same, lower, or higher level of temporal granularity. In practice, COGNET keeps the two processes at the same level of granularity. This is consistent with the large body of literature (c.f., chapter 2 of Card, Moran, and Newell, 1983) that shows these two processes as operating on the same time scale.

#### *2.1.2.7 Multiple Task Instance Principle*

This next principle deals with the abstract nature of cognitive tasks, and further details the relationship between procedural knowledge and declarative knowledge in COGNET. The main theme of this principle is that procedural knowledge may be defined in such a way that it operates on specific pieces of information in memory (called the scope), and that those pieces of information may be defined more abstractly within the cognitive task than they exist in memory. Specifically, the principle implies that items of information in memory may be specific instances of more general concepts or relationships (i.e., because they can exist in multiple instances), and that the items of information in memory may be represented in the cognitive task at this more abstract level. When this is the case, an *instantiation* process is required at the time the cognitive task is activated. This is a process of creating a specific instance of the cognitive task and associating it with a specific instance of information in memory on which it will operate. Thus, a chunk of procedural knowledge that is defined this way, (i.e., in terms of abstract specifications of information, specific instances of which may occur in memory), can be activated multiple times, either sequentially or simultaneously. Each of these activations is an instance of the cognitive task, and is bound to the specific instance of information in memory on which it will operate. The principle also indicates that these task instances, even though they all contain the same procedural knowledge chunk, are separate cognitive tasks from the perspective of the attention process and all other principles in Table 2-1.

This principle also provides some detail, although implicitly, on the organization of declarative knowledge in memory. Specifically, it implies that declarative knowledge can be structured hierarchically with at least two levels of abstraction. The lower level of abstraction is that level at which specific instances or declarative knowledge elements are placed in memory.

The higher level of abstraction allows the same procedural knowledge to be applied to different instances of declarative knowledge in different contexts or multiple instantiations.

#### *2.1.2.8 Task Suspension Principle*

This final principle adds one final state of cognitive tasks, a suspended state that results from ceding the focus of attention on a volitional basis. This principle defines the ability of the cognitive task to place itself in a suspended state and give up the focus of attention, while establishing a condition under which it will become re-activated and again compete to complete execution. The suspended state is in some ways like the inactive state, because a suspended cognitive task is not competing for attention and is awaiting some future state of memory in which a specific pattern is satisfied. Unlike an inactive task, however, the pattern here is not the overall trigger but rather a situational-specific pattern called the resumption condition. In other ways, the suspended cognitive task is like an interrupted task, because it has already had the focus of attention, executed to some internal point, and will continue forward from that point once (or if) it regains the focus of attention. In practice the task suspension principle deals with chunks of procedural knowledge that are constrained by physical embodiment issues. For example, a thread of reasoning about a radar track may be highly chunked (and thus activated as a single cognitive task) but may incorporate points where the result of some external test or communication is required. In such cases, the cognitive task would be suspended until the needed information is in memory, at which time the process could continue.

### 2.1.3 CGF-COGNET Information Processing Mechanisms and Architecture

The information processing mechanisms within the COGNET framework is shown in Figure 2-2. It shows the sensory/perceptual, cognitive, memory, and motor processes,

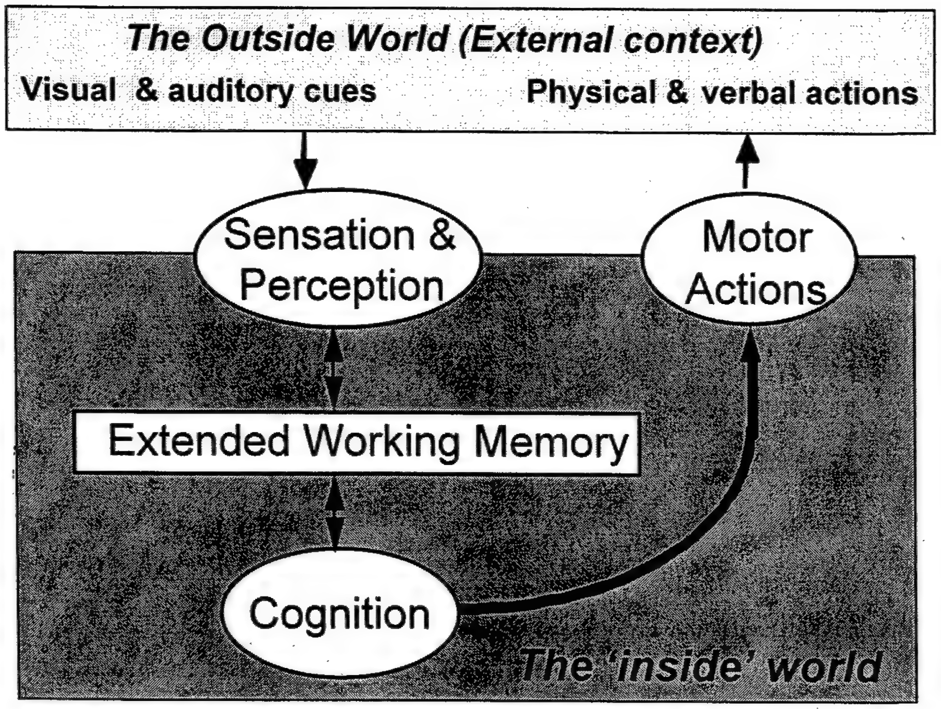


**Figure 2-2. COGNET Information Processing Architecture**

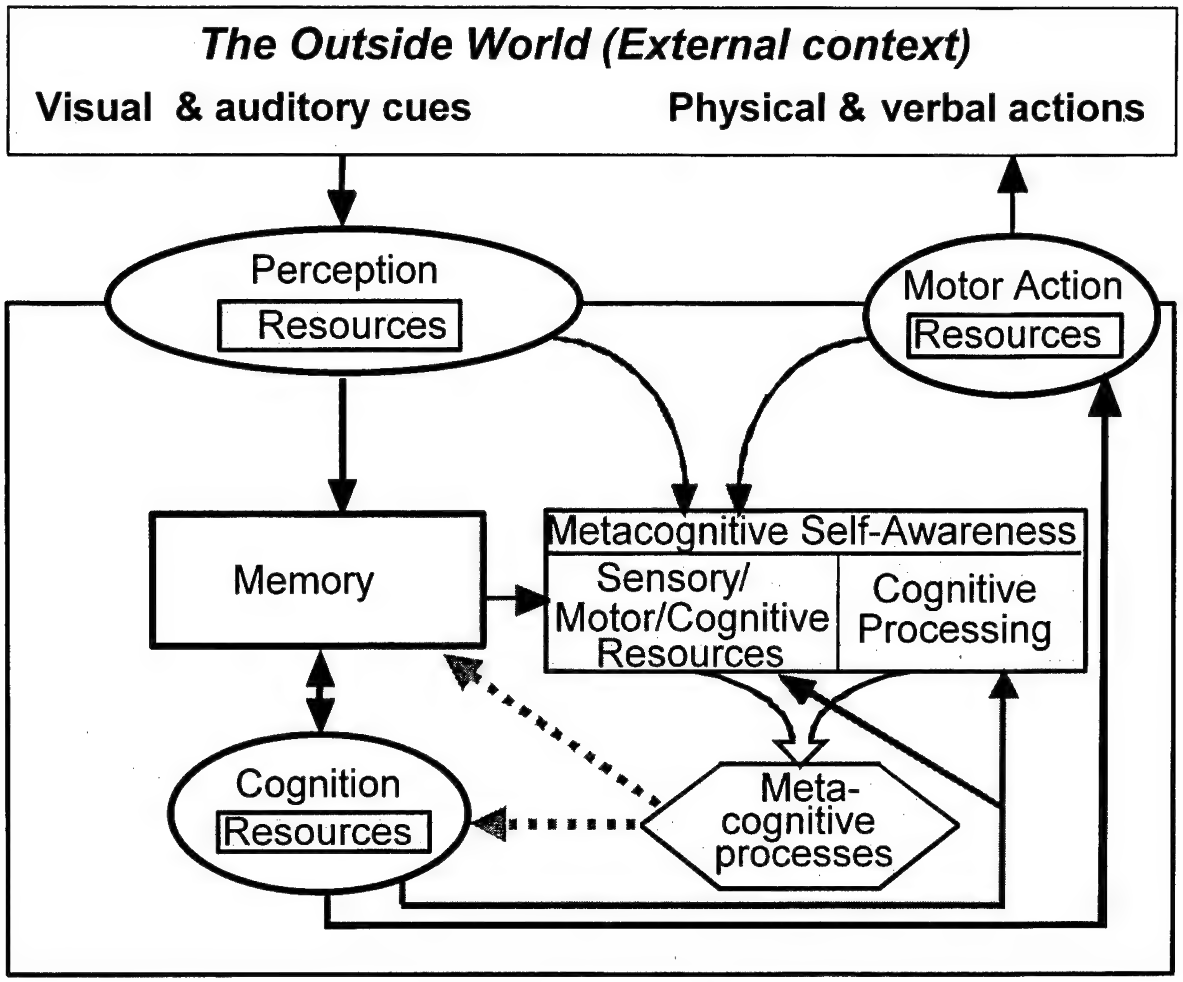


**Figure 2-3. CGF-COGNET Information Processing Architecture**

interconnected as described above. However, also as noted earlier, the AMBR model required a variant of this architecture more oriented toward performance-level models. This is the CGF-COGNET architecture, which is pictured in Figure 2-3.

The CGF-COGNET architecture builds on the basic COGNET architecture, by adding two major types of components, sensory/motor resources (which enable the simulation of time/accuracy constraints on physical interaction with the environment) and metacognitive components (which enable more realistic management of both cognitive and sensory-motor resources). These capabilities of CGF-COGNET are summarized in the subsections that follow.

*2.1.3.1 Sensory-Motor Resources and Time-Accuracy Constraints*

CGF-COGNET extends the information processing mechanisms in COGNET to support the representation and simulation of the time/accuracy aspects of sensory and motor system performance, in four primary ways.

The first extension was to allow the creation of specific resources in each of the processing systems, but with particular emphasis in the motor-action and perceptual system. Rather than pre-define specific resources at a fixed level of granularity, as for example does EPIC, the CGF-

COGNET was designed to allow specific resources to be defined at a level that is *appropriate for the purposes of the specific model being built.* Resources can be defined to have attributes that allow them to be controlled. For example, eyes may have a point of gaze attribute, by which the eyes can be directed; that is, a deliberate eye-movement can be represented as replacing a current point of gaze with a new one. These attributes may also deal with the status of the resources, such as the 'current business' of a hand, or current use of the voice to complete an utterance. The ability to define resources allows CGF-COGNET models to be constrained with human-like limitations, in contrast to the undifferentiated (and unconstrained) sensory and action capabilities in the standard COGNET.[3]

The second extension involved a modification of the architecture to permit multiple, parallel execution threads in each of the three subsystems of the architecture (cognitive, sensory/perceptual, and motor). This allows independent processing activities to be executed in association with the different resources that could now be defined within a given subsystem. For example, the motor system could control separate action process associated with a right hand, left hand, and voice, or the perceptual system could receive sensory inputs from separate visual and auditory processes. The standard COGNET architecture, in contrast, permitted only one thread of activity in each of the main processing subsystems. In addition, CGF-COGNET allows some of the threads of activity in the sensory/motor subsystems to operate either in parallel with cognitive processes or linked with them. This allows, for example, a cognitive process to directly control an on-going motor process or to initiate it for 'ballistic' execution and then proceed in parallel.

The third extension gave the ability to actually control the consumption of time on an execution thread. This enabled a thread of activity (and any associated resources) to become engaged in processes that occur over a period of time. The action processes in the conventional COGNET framework, in contrast, occur as events only, with no inherent time-extensiveness.

The fourth and final extension involved creation of a micromodel construct. This construct allows context-sensitive invocation of a low-level model of the time and/or accuracy of a specific intended activity (motor or sensory) along any execution thread. The micromodel construct also enables the representation of moderators such as stress and fatigue (based on invocation context), as well as individual differences in performance.

#### *2.1.3.2 Metacognitive Capabilities and the Management of Processing*

CGF-COGNET also extends the cognitive architecture of COGNET to incorporate metacognitive capabilities. The term 'metacognition' in CGF-COGNET covers a range of functionality that:

- gives the system a symbolic awareness of the state of its internal information processing,
- provides the system with mechanisms to deal in a more complex manner with interruptions and conflicts among resources, and

[3] Of course, human performance could be and was represented using the standard COGNET representation. It often required, however, additional modeling effort and detracted from the interpretability of the resulting model.

- enables the system to control the flow of reasoning within the cognitive subsystem based on features that are outside the scope of the COGNET Principles of Operation, such as temporal constraints or team/organizational needs.

These functions are discussed in more detail in Zachary, Le Mentec and Glenn (2000), and are summarized below.

Self awareness of resources and processes refers to the ability of CGF-COGNET to maintain an explicit symbolic representation of the cognitive processes being executed, of their execution status, and of the status (and plans for use of) various information processing resources that the current and planned (first order) cognitive processes will require. Such self-awareness is a necessary condition for cognitive models to be able to intentionally modify these processes. It is also necessary for effective self-explanation. The self-awareness is achieved with two extensions to the general COGNET framework. The first is an instrumentation of the information processing mechanisms, including the resources that are defined for a specific model. This instrumentation continuously gathers information on the status of all declared resources and their attributes, as well as on the knowledge being used in all processing subsystems. For the cognitive subsystem, this information includes the status of all cognitive tasks, which are either:

- inactive,
- active but not having the focus of attention,
- active and executing (having the focus of attention),
- interrupted (but still active), or
- suspended.

The symbolic information created by this instrumentation is then made available to the information processing system through the second extension, a declarative metacognitive portion of memory that contains this information.

Interruption management and conflict management refer to the ability of CGF-COGNET to deal with various types of real and potential disruptions to its ability to act purposively. Potential disruptions may arise from two sources: conflicts stemming from the use of or need for specific resources, and conflicts that result from the interruption and resumption of cognitive tasks by the cognitive processor. For example, a cognitive task may be executing a line of reasoning about a specific object such as a radar track, triggered by its relationship (e.g., proximity) to another track. This task could be interrupted by some other more pressing activity, and when it resumes execution, the underlying relationship on which it was predicated may be fundamentally different. In the example, the two tracks may no longer be closing on each other but may now be moving apart. In such a case, continuing with conflict avoidance reasoning would be inappropriate. Detecting the existence of such changes in the mental model of the world is quite difficult; yet failing to do so dramatically degrades the quality and realism of the reasoning process model. Another type of conflict can arise because of the potential for multiple threads of independent activity. If one thread of motor activity is executing a complex motor task, it may intend to begin using one hand and later shift the focus to another. However, a second, independent thread may have already begun execution using the other. Detecting such a conflict is similarly difficult. And in this case as well as the previous task-interruption case, detecting the conflict is only half the problem. Once detected, a way of resolving the conflict must be generated as well.

The second set of metacognitive extensions in CGF-COGNET provide mechanisms to deal with these conflict detection and resolution processes. These extensions build on the first set of metacognitive extensions, specifically on the self-awareness, which provides the basis for detecting conflicts. The conflict management functionality is accomplished through the introduction of procedural knowledge that is purely metacognitive in nature, in that its main purpose is to control the execution of first order cognitive processes, primarily by detecting and avoiding conflicts. These metacognitive procedural knowledge chunks are termed controls, to differentiate them from cognitive tasks. Controls can be triggered in a variety of ways, based on self-awareness information and possibly other information in memory as well. The types of controls and triggering conditions include

- deadlock controls, which are triggered when two threads of activity are contending for a resource and the contention is causing each to be 'locked out', and which when triggered resolve the deadlock according to the procedural knowledge they contain;
- proactive controls, which are triggered by some potential conflict such as an expectation of insufficient time to perform a cognitive task, and which modify execution of the task in some way as to attempt to avoid the conflict;
- interruption/resumption controls, which are triggered when a specific cognitive task is about to be interrupted or resumed, and which can alter the processing of the task to accommodate a smoother interruption (e.g., by forcing completion of some activity or reasoning in process) or a smoother resumption (e.g., by detecting changed information which may affect task processing, and by then determining how the change is to be accommodated).

These various controls compete for the focus of attention as do first order cognitive tasks, but in general have higher intrinsic priority. The procedural knowledge incorporated into controls is a superset of that which can be incorporated into cognitive tasks. Specifically, controls have access to the self-awareness information in the metacognitive memory, and they are able to execute those metacognitive operators that are extensions of the normal COGNET operator set. Metacognitive operators are able to manipulate the aspects of the metacognitive memory that are associated with the flow of attention among the first order processes, such as the priority of a specific cognitive task. This allows metacognitive controls to effectively manage the flow of execution as a way of resolving resource conflict and/or interruption-driven conflicts.

## 2.2 COGNET Expertise Framework

The second major component within the COGNET framework is the representation of expertise. This is the internal information that is processed and manipulated by the information processing mechanisms. Not surprisingly, the types and overall structure of expertise in COGNET are largely defined by the Principles of Operation of those information processing mechanisms (2.1.1 above). COGNET decomposes internal information into four basic types of expertise:

- *declarative expertise* – units of knowledge which contain state/attribute information about (i.e., describe) the external environment, the problem/situation being addressed by the system, and the problem-solving process itself;

- *procedural expertise* – units of knowledge that define teleological states (e.g., goals) and information manipulations (e.g., inferences, physical actions) that can achieve those states;
- *action expertise* – the units of knowledge that define transactions of the motor system that can be used to implement intended effects/actions in the external environment;
- *perceptual expertise* – the units of knowledge that defines processing operations to generate/transform internal information in response to information that is sensed from the external environment.

This decomposition is maintained in CGF-COGNET, with one addition:

- *metacognitive expertise* – the units of knowledge used to control the selection and execution of procedural knowledge

In COGNET, the only types of metacognitive knowledge are the triggers and priority measures of cognitive tasks, and they are actually incorporated into the cognitive task itself. However in CGF-COGNET, as discussed above, there are separate metacognitive mechanisms and thus separate metacognitive expertise components. Specifically, there is the declarative metacognitive memory (i.e., self-awareness), and procedural metacognitive knowledge (i.e., the various metacognitive controls and operators).

In terms of the COGNET and CGF-COGNET architectures:

- declarative knowledge, including declarative metacognitive knowledge, is maintained in memory, and modified by both perceptual and cognitive processes,
- procedural knowledge is executed by the cognitive process, and both manipulates information in (declarative) memory (excluding the metacognitive portions), and sends intended actions to the motor system.
- action knowledge is processed by the action/motor system, and manipulates the external environment.
- perceptual knowledge is executed by the perceptual process as information is sensed in the external environment. As the perceptual knowledge is executed, it manipulates information in the (declarative) memory.
- metacognitive procedural knowledge is executed by the cognitive process, and manipulates declarative memory (including the metacognitive portions).

The overall strategy for representation of each of these types of expertise is driven by the focus of the overall COGNET system as discussed earlier – on expert-level competence (or performance, in the case of CGF-COGNET) in complex real-time environments. Theories of expertise and skill acquisition clearly point to the fact that experts rely, within their domain of expertise, on rich and highly compiled knowledge structures that have chunked many lower level productions into contingent structures that minimize the search of the knowledge space. In this view, specific desired end-states, often called goals, are matched at a very high level with features of the situation to call up a prepackaged, albeit abstract, strategy. The strategy is then instantiated in terms of the specific details of the problem/situation, and executed.

This view is strongly evident in the COGNET Principles of Operation, and as noted in that discussion, it has implications for the representation of both declarative and procedural knowledge. For declarative knowledge, it implies that there needs to be some hierarchy of abstraction in representation of the declarative information, to support the pattern-matching and instantiation processes, which work at differing levels of abstraction. For procedural knowledge, it implies that there is some internal contingent structure in which the application of lower-level units is adapted to the specifics of the current instance.

The strategy used in developing COGNET was to identify well worked-out formal representation schemes that could be adapted to reflect the above perspective. For procedural knowledge, this was done by adapting and extending the hierarchical goal decomposition notation called GOMS (Goals-Operators-Methods-Selection Rules) first developed by Card, Moran and Newell (1983). GOMS was in fact developed as a way of capturing human-computer interaction competence. For declarative knowledge, this was done by adapting and extending the hierarchical knowledge representation called the blackboard representation (c.f., Nii, 1986a,b). The blackboard representation was originally developed to capture the multiple levels of declarative information used by human in the process of understanding spoken speech, in a speech understanding system called Hearsay (Erman et al., 1980). Although it was later widely used in Artificial Intelligence (see Carver and Lesser, 1994), B. Hayes-Roth (1985) among others have demonstrated its ability to capture the kinds of declarative expertise used by people in complex tasks.

The requirements for the representation of perceptual knowledge were largely defined by the Principles of Operation, which specified that this type of knowledge was self-activated and essentially unitary in nature. The basic framework for this was first established by Reiger (1977) as a 'spontaneous computation' knowledge source, to which in fact the term demon was applied. This representation has been subsequently evolved and applied by others, and formed the basis for the definition of perceptual knowledge in COGNET. The various influences that shaped the COGNET representation of expertise are summarized in Figure 2-4. The specific representation

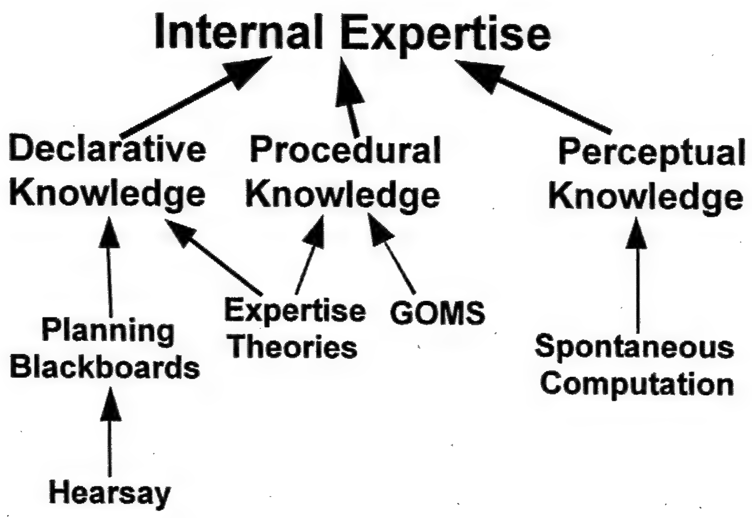


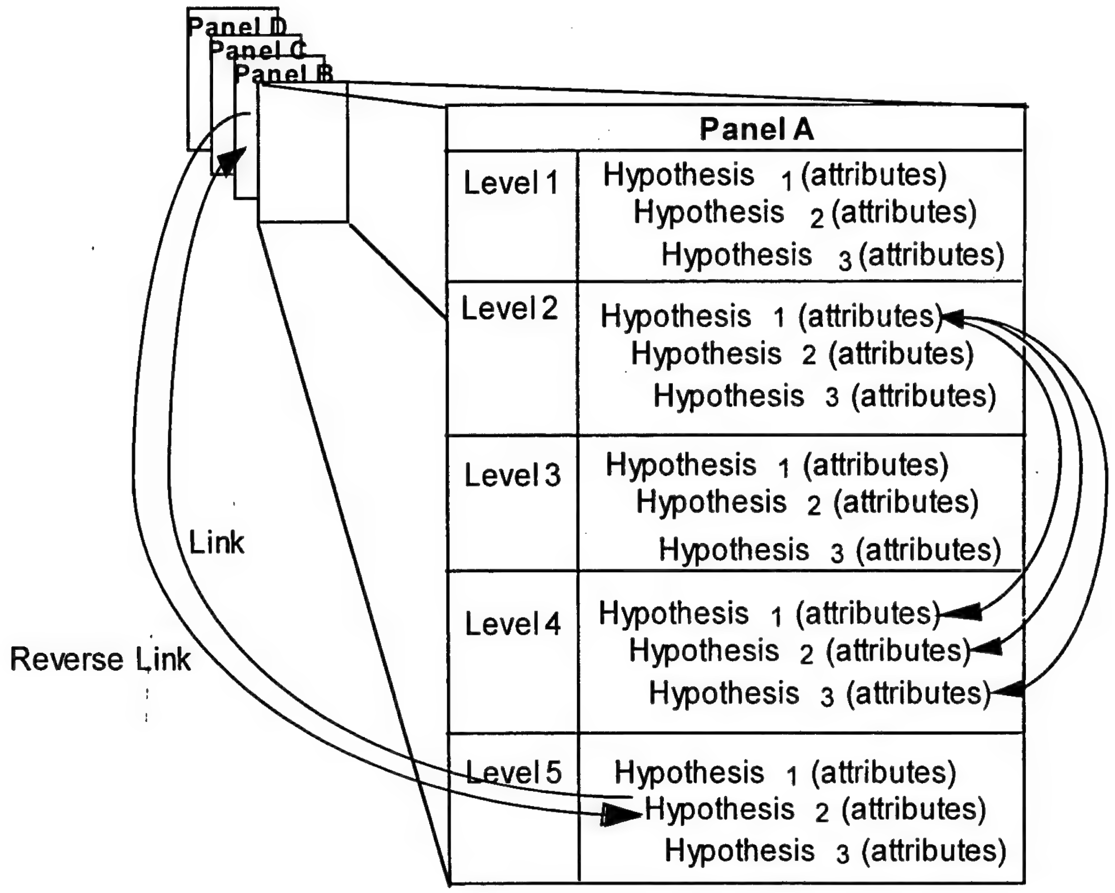
**Figure 2-4. Influences in COGNET Expertise Representation**

used in COGNET (and CGF-COGNET where it differs) for each of these types of expertise is discussed below. After this, the relationship with the external environment in COGNET is discussed

2.2.1 Declarative Expertise

Expertise in declarative memory in COGNET (and CGF-COGNET) is represented in a blackboard format. In this representation, information is accessed for storage, retrieval and modification purposes in units that are called hypotheses. The hypothesis represents a single concept, but is not itself the smallest unit of declarative information, as hypotheses may contain attributes, as well as bi-directional associations or links with other hypotheses

As discussed above, the Principles of Operation require these units of declarative information in memory (i.e., hypotheses) to be represented at two hierarchical levels of abstraction (at least). The blackboard representation in COGNET accomplishes this with a panel/level structure, in which information in memory is organized into abstract groupings called panels, each of which is organized into less abstract units called levels. Each level defines a specific class of hypothesis, and hypotheses of that class are accessed through that specific panel/level. This structure is shown in Figure 2-5.



Panel D
Panel C
Panel B
Panel A
Level 1
Hypothesis 1 (attributes)
Hypothesis 2 (attributes)
Hypothesis 3 (attributes)
Level 2
Hypothesis 1 (attributes)
Hypothesis 2 (attributes)
Hypothesis 3 (attributes)
Level 3
Hypothesis 1 (attributes)
Hypothesis 2 (attributes)
Hypothesis 3 (attributes)
Level 4
Hypothesis 1 (attributes)
Hypothesis 2 (attributes)
Hypothesis 3 (attributes)
Level 5
Hypothesis 1 (attributes)
Hypothesis 2 (attributes)
Hypothesis 3 (attributes)
Link
Reverse Link

**Figure 2-5 Blackboard Representation of Declarative Information in COGNET**

Formally, the blackboard is described as a series of independent *panels*:

$$\text{Blackboard} = \{\text{Panel}_1, \text{Panel}_2, \ldots \text{Panel}_n\}$$

Each panel is then described as a series of hierarchical *levels*:

$$\text{Panel}_i = \{\text{Level}_1, \text{Level}_2, \text{Level}_3, \ldots \text{Level}_m\}$$

Items of declarative knowledge are contained in hypotheses that are accessed via a specific panel and level of this blackboard structure. Each hypotheses type is defined by its panel/level, as containing a predefined number of attributes:

$$\text{Panel}_i, \text{Level}_j = \{\text{Object}_k, \text{with attributes Attribute}_1, \text{Attribute}_2, \ldots \text{Attribute}_p\}$$

A hypothesis type need not contain any attributes, and attributes may be any type of information – numeric, symbolic, etc. Specific instances of this type, i.e., individual hypotheses, are created (or POSTed as it is termed in COGNET), with no values specified for any attributes. The POSTing is done, as specified in the Principles of Operation, through cognitive operators within cognitive tasks and perceptual demons (which are discussed below).

Individual hypotheses, in COGNET, can be linked with other hypotheses by means of cognitive operators or perceptual demons. These links can represent semantic relationships or experiential associations. Although in principle any hypothesis can be linked to any other hypothesis, actual links between specific hypotheses must be created as instances of link-types, which are defined as possible associations between different hypothesis-types (i.e., panel/level pairs). These link-types are defined simply by naming them and defining the panels and levels they associate:

$$\text{Link-type}_i = ((\text{Panel}_{i1}, \text{Level}_{j1}), (\text{Panel}_{i2}, \text{Level}_{j2}))$$

Specific links between specific hypotheses are created through cognitive operators within cognitive tasks and perceptual demons.

### 2.2.2 Procedural Expertise

The general representation of procedural expertise is defined by the Principles of Operation, which specify that it be organized into highly chunked units termed cognitive tasks. These cognitive tasks have essentially two parts, a metacognitive 'wrapper', which contains the trigger and priority measure, and the body, which contains the actual procedural knowledge.

The task body is represented as a hierarchy of contingent goals and cognitive operators, structured in a format similar to that used in the GOMS notation originated by Card, Moran and Newell (1983) and since widely modified and applied by others. However, the COGNET cognitive task description includes additional features to allow for:

- accessing, creating, and manipulating information in the declarative memory blackboard,
- constraining activation of lower level goals and methods on the basis of patterns of information in the declarative memory blackboard,

- instantiation of both behavioral (i.e., action-) operators and cognitive operators in context-based ways, based on the declarative memory blackboard, and
- interruption, suspension, and subrogation of the current task.

The core notation for describing the knowledge in the body of cognitive tasks is summarized in Table 2-2.

### 2.2.3 Action Expertise

Action knowledge is implied but not specifically incorporated into the COGNET Principles of Operation. In COGNET, action knowledge in the motor system is invoked by the Perform Action operator. This operator activates and executes an instance of the type of action knowledge named in the Perform Action operator. The information from declarative memory that is incorporated in the operator allows the action system to appropriately instantiate the requested action knowledge.

In CGF-COGNET, the representation may also incorporate additional elements that allow for ballistic action sequences (see 2.1.2.1) and that allow the performance time and/or performance accuracy of the resulting action to be predicted. Time/accuracy predictions are created by a construct called the Micromodel, which in structure is similar to a CALCULATE operator (see Table 2-2). The Micromodel invokes a separately constructed model that predicts a time or accuracy parameter for an action about to be invoked. Obviously, the Micromodel may also utilize information from declarative memory in its estimation process. For example, if the action desired is to push a button, a Micromodel of hand movement may be executed, e.g. which encodes a version of Fitts' Law (Fitts, 1954) to predict the time of the movement, given self-awareness of where the hand is now and internal declarative knowledge of where the button is. A different Micromodel could be invoked to estimate the accuracy of the action. The estimate(s) produced by the Micromodel can then be used as a parameter of a SPEND_TIME operator, which causes a specified amount of time to be consumed on that activity thread before the next operator can be executed.

### 2.2.4 Perceptual Expertise

Perceptual expertise is represented in a simple manner in COGNET, using variants of the procedural POST/UNPOST/TRANSFORM cognitive operators (see Table 2-2). The basic unit of the perceptual expertise is the perceptual demon, as defined in the Principles of Operation. Each perceptual demon consists of two parts, an environmental cue definition and an associated memory modification operator (i.e., POST/UNPOST/TRANSFORM). The environmental cues represent types of information that can be sensed, or patterns of such types. Consistent with the spontaneous computation representation of perceptual knowledge, when any information is sensed that matches any demon's pattern, the demon is activated and its memory modification operator is executed. This results in some hypothesis in declarative memory being created, modified or deleted as a result of the perceptual processing of the sensed information.

In CGF-COGNET, a separate filter may be constructed between the external cues as sensed and the perceptual subsystem where the demons are activated and executed. This filter represents limitations imposed by the sensory resources, such as field of view or foveal diameter, and necessarily relies on the cognitive proprioception of the sensory resources involved. For example, field of view limitations rely on the self-awareness of the eye's momentary point-of-gaze. In addition, Micromodel and SPEND_TIME operators can be incorporated into perceptual demons in CGF-COGNET to represent time-extensive perceptual processes, e.g., reading a line of text, picking a specific object out of a field of objects, etc.

**Table 2-2. COGNET Description Language for Procedural Expertise**

**TASK: TASK-GOAL-NAME, *trigger, priority formula***

**(procedural body)**

= definition of a cognitive task, which is named for the overall goal which the chunk of procedural knowledge represents. The trigger and priority formula form the metacognitive information which determines, in combination with the momentary contents of the declarative memory blackboard, when and if the cognitive task is activated and obtains the focus of attention. Once it has the focus of attention, the procedural body is executed. The procedural body consists of one or more goals and cognitive operators.

**Elements of Task/Goal/Method Procedural Body:**

**GOAL: GOAL NAME...<...*condition*>** = initiates execution of a subtree called the GOAL's body, which includes one or more operators and possibly lower level subGOALs

**OPERATOR** is any of the following

<u>**POST:Panel:level:{attribute-name, attribute-value}**</u> = creates a new hypothesis at the specified level of the specified panel, and assigns values to the attributes as specified

<u>**UNPOST hypotheses-reference**</u> = unposts the hypothesis identified by the hypothesis reference

<u>**TRANSFORM hypotheses-reference,{attribute-name, attribute-value}**</u> = transforms the specified values of the specified attributes of the hypothesis identified in the hypothesis reference

<u>**DETERMINE operation-name ({data/ parameters})**</u> = execute an instance of the named generic symbolic operator; returns a numeric result from a separately defined operation, with optional parameters (which may include performance time in the CGF-COGNET version only)

<u>**CALCULATE operation-name({data/ parameters})**</u> = execute an instance of the named generic mental- arithmetic operator; returns a numeric result from a separately defined operation, with optional parameters (which may include performance time in the CGF-COGNET version only)

<u>**Suspend until [condition or future time]**</u> = suspends the current cognitive task, setting a condition for resumption or a future time for resumption

<u>**Perform ACTION ({data/ parameters})**</u> = invokes the motor system to perform ACTION using the specified parameters

<u>**Use Method ({parameters})**</u> = invoke execution within the current cognitive task of a context-independent chunk of procedural expertise to be instantiated using the specific parameter values provided

Embedded terms are defined as given below

<u>***trigger***</u> is a Boolean statement based on blackboard hypotheses, including multi-level existential quantifiers, which will activate the cognitive task whenever it evaluates to True

***priority formula*** is a numerical formula based on blackboard hypotheses and/or constants, including multi-level existential quantifiers, whose value at any point in time will define the priority of that task for capturing the focus of attention

***condition*** is either

a Boolean statement based on blackboard hypotheses, including multi-level existential quantifiers

OR

context free CONTROL information, such as *repeat until ..., continue while, etc*

***hypothesis-reference*** is either

a **FIND** operator with an associated Boolean expression based on blackboard hypotheses including multi-level existential quantifiers; the hypotheses that satisfy the expression are the referents of the FIND

OR

a name or pointer that is associated with a specific hypothesis in memory (the association can be established when the hypothesis is POSTed or when a FIND is executed)

A METHOD, invoked by the USE METHOD operator, is a separate chunk of procedural knowledge, and is defined as follows. Its procedural body is constructed in the same way as the procedural body of a Cognitive Task

**METHOD method-name: <*parameters*>**

(procedural body)

### 2.2.5 Metacognitive Expertise

There are three types of metacognitive expertise in CGF-COGNET. The first is the metacognitive trigger and priority formula associated with each procedural knowledge chunk, common to COGNET and CGF-COGNET, and previously discussed (2.2.2). The other two exist only in the CGF-COGNET variant of the system. These are self-awareness declarative memory, and metacognitive controls, which were discussed in 2.1.2.2 above. The representation used for each is detailed below.

#### *2.2.5.1 Declarative Metacognitive Memory – Self-awareness*

Declarative metacognitive memory in CGF-COGNET is represented in the same was as other types of declarative memory, i.e., using a blackboard representation. In CGF-COGNET, there is a separate metacognitive blackboard on which the self-awareness information from the CGF-COGNET information processing mechanisms is posted. Table 2-3 presents the types of hypotheses that are stored in the generic or default part of the metacognitive blackboard. Only one panel is provided by default. This is named the Task panel, and contains information on current and past task activation. It provides self-awareness of information on current and past cognitive processing at the task level. It consists of five levels, named the Task level, the Instance level, the Current level, the History level and the Ordering level. The panels are interconnected with a number of links, as described below.

**Table 2-3. Organization of the Default Metacognitive Blackboard Panel**

| Panel name | Level name | Attributes | Links/Reverse links |
|---|---|---|---|
| Task | Task | Task Name | Instance/Task |
| Task | Instance | ID<br>Activation time<br>Start time<br>Context<br>Trigger Environment<br># of interruptions<br>Estimated Duration | Current/Instance<br>History/Instance<br>Task/Instance<br>Ordering/Instances |
| Task | Current | Time Spent<br>Status | Instance/Current |
| Task | History | Actual Duration<br>End Time | Instance/History |
| Task | Ordering | | Instances/Ordering |

Additional panels may be created for use in specific models, particularly to capture self-awareness of specific sensory-motor resources that were created for that model. Because of the variable granularity approach CGF-COGNET employs, however, there are no default resources and thus no default self-awareness resource information.

*2.2.5.2 Procedural Metacognitive Expertise – Controls*

A control is a metacognitive procedure that is triggered by the occurrence of a specific event, on either a reactive basis (i.e., the event has occurred) or a proactive basis (i.e., the event may occur in the future). Various types of controls are incorporated in CGF-COGNET to react to different classes of events, as discussed in 2.1.2.2 above. The representation of controls involves two different components:

1) control definition – an explicit representation of the procedural knowledge in the control as a stand-alone definition; and
2) control declaration – a specification of what control should be used, where and under what condition. These specifications are embedded within the cognitive tasks that are affected by the controls.

Controls are defined in a manner analogous to the METHOD and DETERMINE constructs used to represent context-free procedural knowledge (see Table 2-2). That is, the metacognitive procedural knowledge in the control is contained as a procedural body of an operator of the form

**DEFINE_CONTROL <control-type> <control-name> (control-parameters)**

The procedural body is represented using the procedural knowledge operators in Table 2-2, except that the embedded terms may refer to the metacognitive portions of declarative memory as well as the non-metacognitive portions.

A control declaration specifies what control is to be used under what conditions in a cognitive task. The declaration is of the form

**ON <control-type> <control-name> (control-parameters)**

The control-type specifies the type of control (e.g., interruption, recovery, deadlock, etc.), which in turn defines the types of parameters it requires. A deadlock control, for example, needs the hypothesis in the metacognitive blackboard that represents the resource to which the control applies.

A declaration can be placed globally in the model or local to a Task, Goal or Method. A global declaration may mean, for example, that a specific control is always to be used when the event (e.g., a hand-resource deadlock) occurs. A more localized declaration applies only to the lowest level of procedural knowledge (Task, Goal, subGOAL, etc.) in which it is placed, allowing the metacognitive processing to be invoked and applied in a highly context-sensitive manner.

### 2.2.6 Interactions with the External World in COGNET

As should be clear from the preceding discussion, a COGNET and CGF-COGNET model's interaction with the external world depends on both the internal processing mechanisms and the internal expertise. In fact, though, it is the internal expertise that is critical. Although it is the sensory capability that detects external cues, the information registered can only be internalized when there is some procedural knowledge available to internalize information about that cue in memory. Similarly, although it is the motor system that implements action, the overall system can only take those actions about which it possesses appropriate motor knowledge. Thus, without appropriate perceptual knowledge to allow the model to make sense of what it senses, or appropriate action knowledge to allow the model to manipulate the external world in a purposive way, the processing mechanisms are of no utility.

At a deeper level, the finiteness of a specific COGNET/CGF-COGNET model does place limits on its interactions with the environment. The way in which the patterns of demons are expressed, for example, must match precisely with the way in which cues are sensed and registered internal to the system. Even a slight 'impedance mismatch' can result in information being lost or ignored. Similarly, the actions that the motor system attempts to take must match the affordances in the environment. Again, even a slight mismatch can result in actions not being successfully taken. This is clearly an artifact of both the way synthetic cognition systems work, and of software systems in general. They require the physical and data interface between the model and the external world to be engineered in a fairly precise manner. This has been a main concern in the development of the actual software system that implements COGNET, as discussed below in 2.4.

## 2.3 Three Types of COGNET Models

The preceding discussion permits a differentiation between several types of models that can be built with the COGNET framework. The simplest type of model is a representation of the internal expertise that underlies expert-level competence in a given domain. This is an 'expertise model'. When represented in the COGNET description language as discussed above, the expertise model is both formal and executable. However like a program without a processor, its executability is only theoretical. The expertise model can be made practically executable by adding the other two terms in the essential COGNET equation, i.e., processing mechanisms and external context. This involves adding:

- a software emulation of the underlying information processing mechanisms so that the expertise can be executed and applied, and
- a connection to a (real or simulated) external environment, which provides appropriate sensory cues, affords opportunities for physical manipulation of the environment, and connects the two with realistic problem dynamics.

The result is a cognitive model that is in embodied and situated in a context, and is able to simulate fully competent cognitive processing within that context. This model, which subsumes the expertise model, is called a 'competence model'. The relationship between the two is shown in Figure 2-6a and 2-6b.

The behavior of the competence model differs from the behavior of actual human experts in that it is not constrained by many of the factors, which limit people's ability to perform perfectly, such as the time and accuracy terms, which are introduced into the COGNET 'performance' equation. These limiting factors are incorporated by adding to the competence model a set of (typically domain specific) features, which represent the factors that limit performance in that domain. These features can include such things as timing, accuracy, reach limitations to manual actions, limits to visual performance such as acuity and field of view, channel conflicts on auditory channels, and so on. Consistent with the overall philosophy of COGNET, these factors are not organic to the system. Rather, they are constructed and incorporated as needed for specific applications (returning again to the notion that a model is a representation constructed for a specific purpose). This allows for flexible granularity and scope of features in the representation process. These limiting factors thus envelop and constrain the competence model, and yield a performance model, as shown in Figure 2-6c.

## 2.4 Software Support for the Model-building Process

The development of these three classes of COGNET models obviously requires various types of software support, beginning with the software emulation of the internal processing mechanisms. Over the last five years, the COGNET research team at CHI Systems has developed an entire software environment to support the building, execution, testing and application of the different types of COGNET models. This environment is named iGEN™, and consists of several components. The main component is the software engine that emulates the internal processing mechanisms and functions according to the Principles of Operation discussed previously. This engine is called BATON (Blackboard Architecture for Task-Oriented

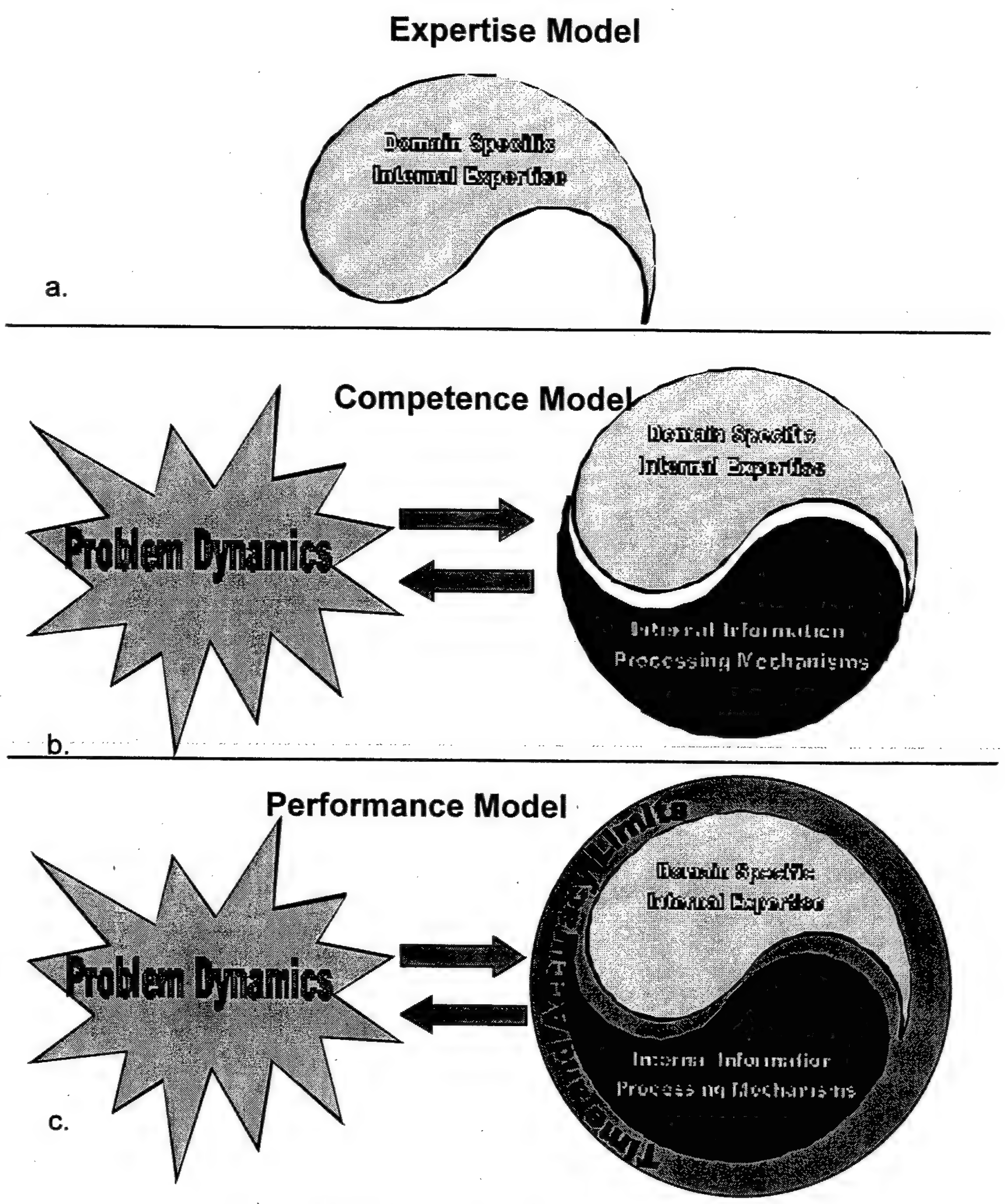

Expertise Model
Domain Specific Internal Expertise
a.
Competence Model
Problem Dynamics
Domain Specific Internal Expertise
Internal Information Processing Mechanisms
b.
Performance Model
Problem Dynamics
Time/Accuracy Limits
Domain Specific Internal Expertise
Internal Information Processing Mechanisms
c.

**Figure 2-6. Three Types of COGNET Models**

Networks). As can be easily seen in Figure 2-7, BATON executes some body of internal expertise, i.e., expertise model, within a specific problem context. The expertise model is represented in two different forms in iGEN™. BATON itself operates on a highly formal representation of the COGNET expertise description language that is embedded in a list-processing syntax and is called the COGNET Execution Language or CEL. While CEL can certainly be read and authored by people, it does require some substantial programming skill.

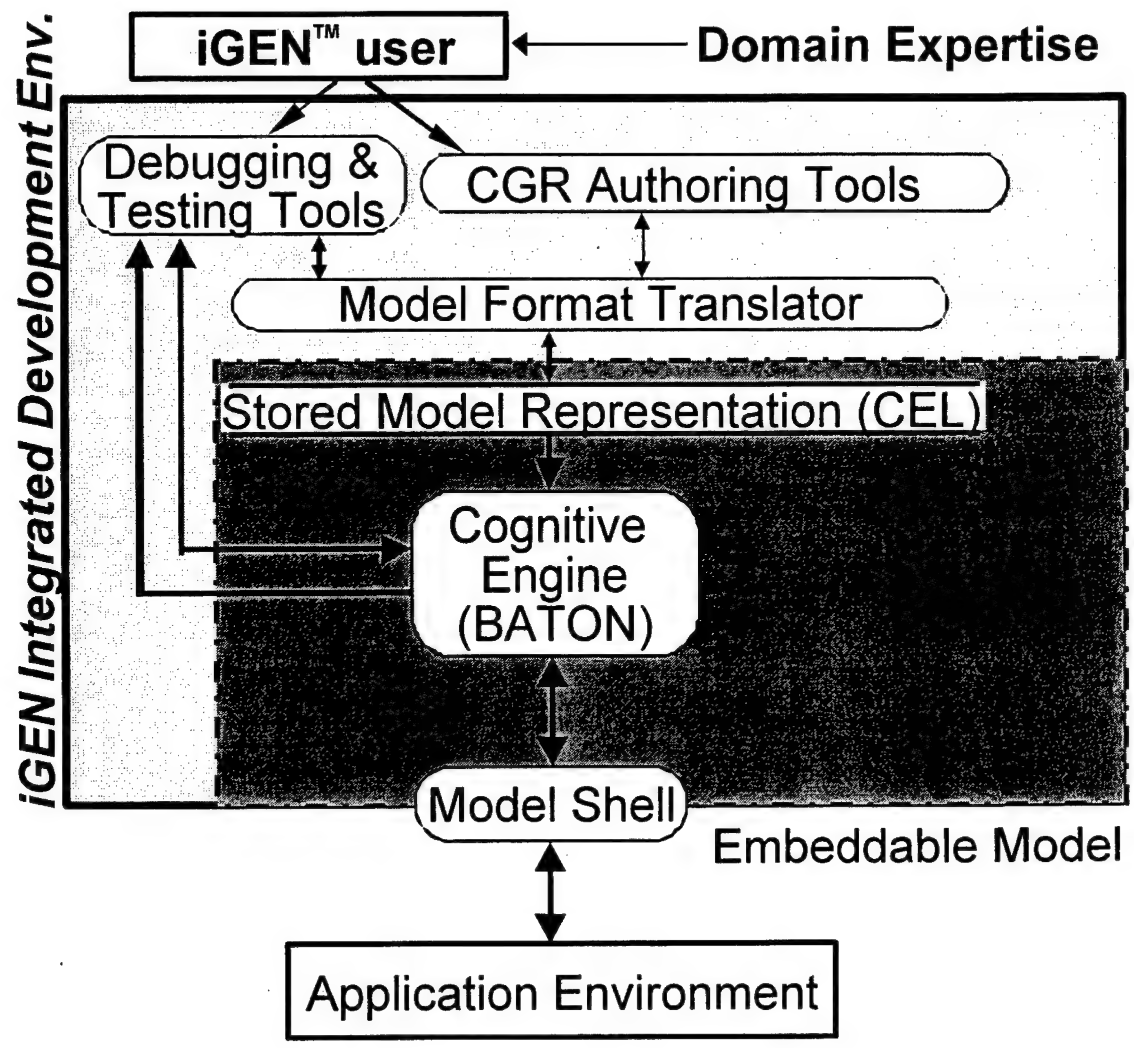
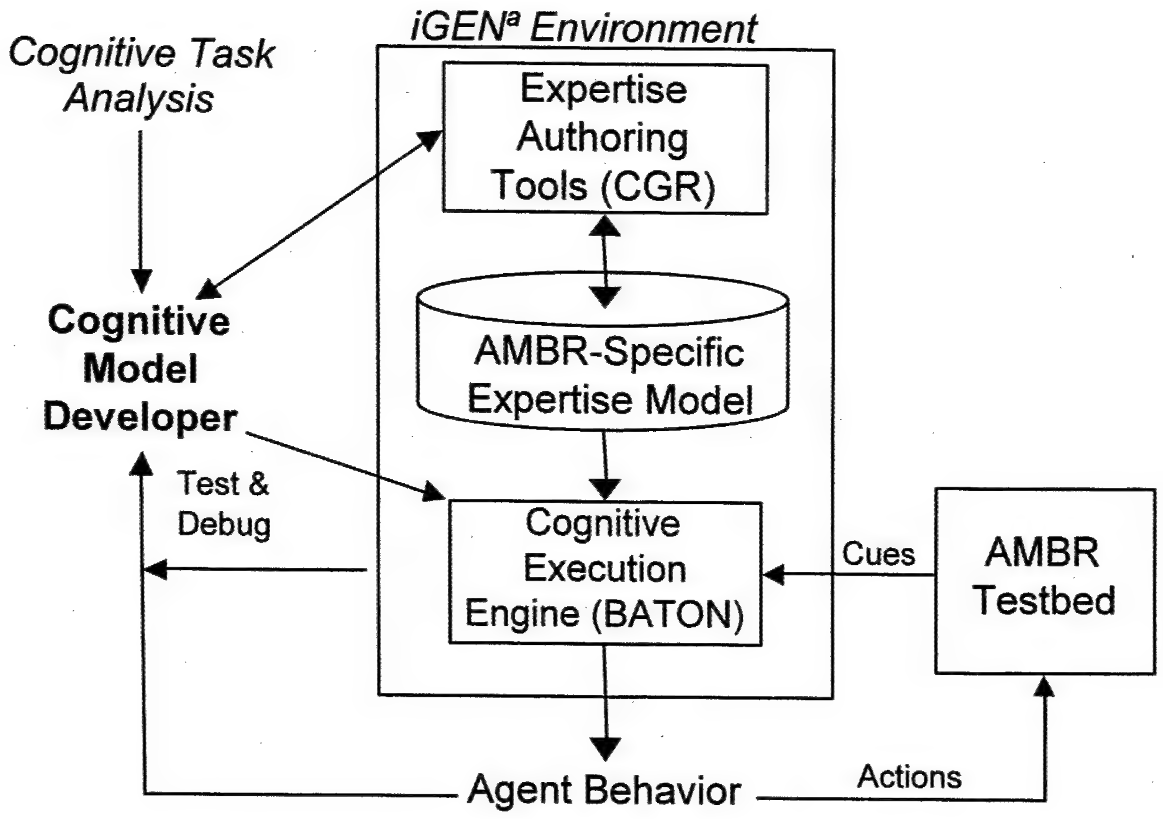


**Figure 2-7. Organization of the iGEN™ Cognitive Agent Toolkit**

To reduce this need for programming knowledge, a graphical or visual programming interface to CEL was created within iGEN™. This is the CEL Graphical Representation or CGR, and is the primary means by which iGEN™ users interact with the expertise model. The translation between CEL and CGR is done automatically and continuously by iGEN™.

iGEN™ also incorporates a set of debugging tools to help the model developer isolate and correct errors in expertise models, and a C++-based Application Program Interface (API) which allows the expertise model and BATON to interact with an external context. iGEN™ assumes that this external context is created or interfaced with separately from iGEN™, although iGEN™ does provide a capability for interactions with an external context to be scripted as an aid to model building. The interface that is built with the API, between a specific expertise model and the context in which it operates, is called the model's 'shell'. Once an expertise model and shell have been fully debugged and tested, it is possible to detach the expertise model (in its CEL form), the BATON engine and the shell from the rest of the iGEN™ development environment.



iGENª Environment
Cognitive Task Analysis
Expertise Authoring Tools (CGR)
Cognitive Model Developer
AMBR-Specific Expertise Model
Test & Debug
Cognitive Execution Engine (BATON)
Cues
AMBR Testbed
Agent Behavior
Actions

**Figure 2-8. Use of iGEN™ to Develop the AMBR ATC Model.**

This creates a separate, standalone or embeddable cognitive model that can be integrated into another software application, such as a CGF simulation, an intelligent tutoring system or a decision support system. The overall structure of iGEN™ is shown in Figure 2-7.

Figure 2-8 depicts the process by which the Air Traffic Control performance model was developed for the AMBR program. The CGR authoring tools were used to formalize the results of a cognitive task analysis (see Section 3 below) into a specific COGNET expertise model of the AMBR task. At the same time, a software shell was constructed that allows the BATON engine to 'see' the ATC testbed displays, and to 'manipulate' the testbed controls (keyboard and mouse). Once the initial expertise model and shell were built, the model developer(s) systematically tested and debugged both, using the iGEN™ debugging tools. When the debugging was finished, the AMBR expertise model, BATON, and AMBR shell were detached from iGEN™, compiled into a stand-alone application, and transported to the moderator site for the fly-off data collection. (It should again be noted here that the versions of BATON and iGEN™ used in AMBR were those based on the CGF-COGNET variants of the COGNET framework, as discussed above, not the commercially available iGEN™ system.)

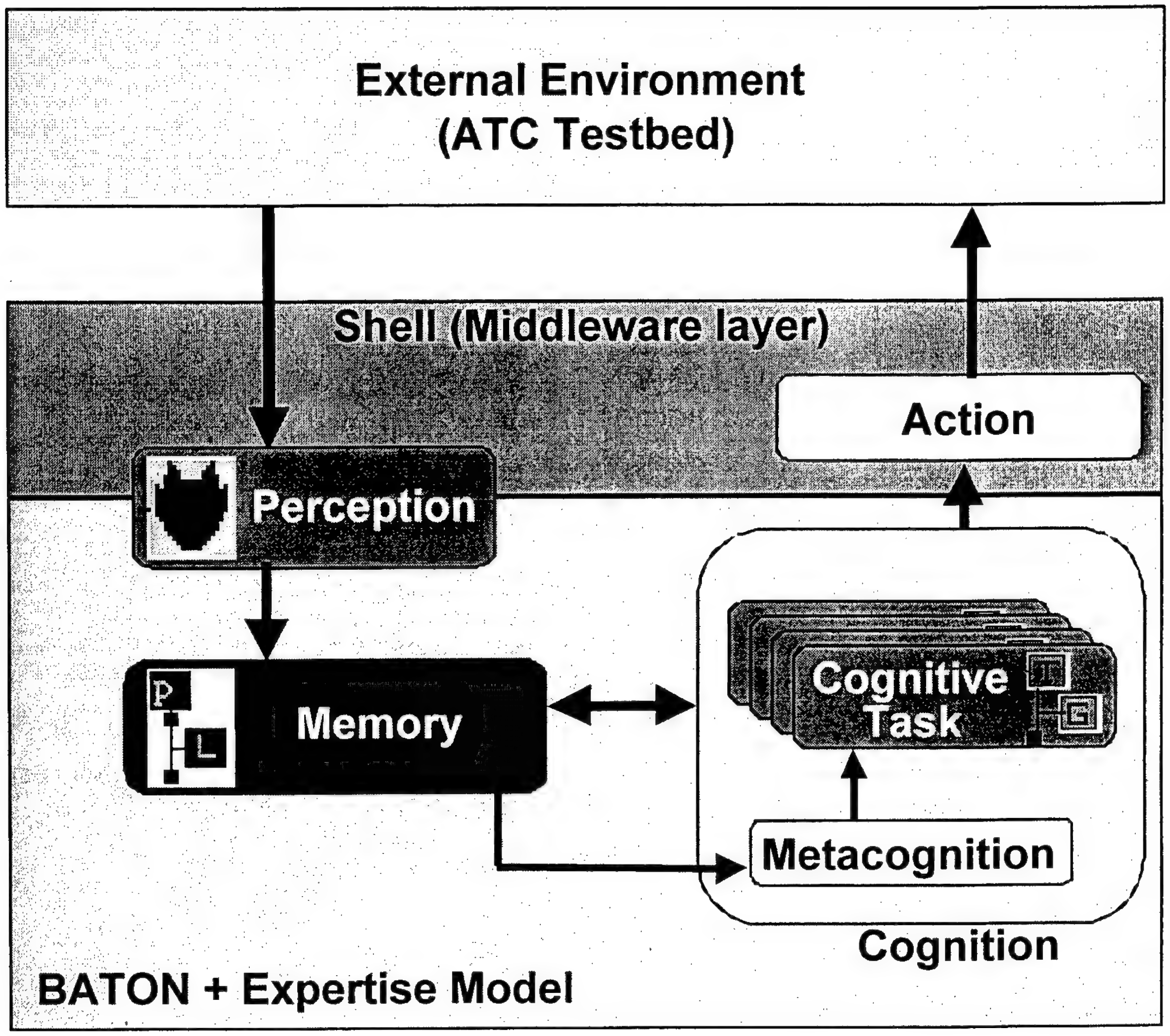


**Figure 2-9. Execution of COGNET/iGEN™ Model**

The process by which the iGEN™ competence or performance model is executed is pictured in Figure 2-9.

The external environment typically exists as a separate software entity, ranging from a simple script to a full simulation environment or even a 'live' operational software environment such as an actual air traffic control system. The model has access to this environment through the shell, which performs two primary functions by providing:

- inputs to sensation – the shell captures specific events or processes within the environment and translates them into cues that will become accessible to the sensory system (in the case of a performance model) or directly into the perceptual system (in the case of a competence model).
- outputs to the environment – the shell translates actions taken by the model, such as hand movements or verbal messages, into software events or processes within the external environment. For example, a hand movement to depress a button

could be translated into a message to depress the button electronically at a specific time, or a verbal message might be translated into a command to a voice synthesizer and/or a message to another software component.

Before an actual application is executed, the BATON executes the initialization component of the expertise model, which populates memory with various elements of declarative knowledge that are to be present in memory when the actual execution starts. Once the execution begins, the three information-processing subsystems (cognitive, perceptual, and motor) in BATON begin to operate in parallel. Cues from the environment are registered by the perceptual subsystem via the shell, and appropriate perceptual demons are activated within BATON, resulting in information being posted on the blackboard structure, which represents memory. Simultaneously, the various cognitive tasks begin to compete for the focus of execution attention in the cognitive processor, with the metacognitive information (triggers, priorities, suspension conditions, etc.) determining where and how the focus of attention flows. Any cognitive task may, while being executed, modify the contents of memory. At various points in time, the action process may be invoked by a cognitive task, defining specific actions to be taken. The action processes, which reside entirely within the shell, generate events in the environment, which implement the requested action.

## 3. MODEL DESCRIPTION

### 3.1 Modeling approach

Our main objective for the AMBR program was to create a model that could perform the ATC task in a human-like manner, generating performance measures, response times, and post-task subjective workload assessments within ranges documented for human subjects. Not only did the model need to generate realistic outputs for given inputs (pass a Turing test), it also had to use a human-like strategy, as indicated by a cognitive task analysis (CTA). The goal of correspondence to human methods or strategies is primarily a practical rather than a theoretical goal, in the sense that realistic performance (which is highly desirable from the perspective of Computer Generated Forces) is more likely to be obtained in complex multi-tasking situations using human-like cognitive processing strategies and internal representations than using other AI techniques. However, it is a theoretical goal to the extent that the rich literature on expertise (e.g., Chi, Glaser & Farr, 1988; Ericsson & Smith, 1991; Hoffman, 1992) informs the structure and flow of the model. An example of this is the incorporation of the notion of 'mental model' in the problem representation blackboard.

We were not concerned with correspondence at the level of specific mechanisms in a physiological sense for modeling either sensory-motor or cognitive aspects of processing. In fact, COGNET is generally neutral with respect to specific theories of sensory-motor processing, allowing the modeler to determine the appropriate approach and level of granularity for a specific modeling problem. In general, our approach is to use "just-enough" granularity to adequately describe the phenomena and generate realistic performance to meet a goal, while expending a minimum of modeling effort. As has been pointed out by Pylyshyn (1989) among others, "computers can enter into the detailed process of constructing models of cognitive processes at several levels… the more fine-grained the correspondence match, the narrower the range of phenomena the model is able to cover." (p. 62). Thus, a coarser grain model can deal with a wider range of phenomena, generally a benefit for modeling complex, real-world task performance.

A number of characteristics of the problem and problem domain shaped our approach to the model. The key aspect of the problem was multi-tasking behavior—prioritizing tasks, shifting attention among tasks, and modification of multi-tasking management strategy due to changes in workload. COGNET was designed to handle multi-tasking problems — the cognitive architecture and Principles of Operation (described in Section 2 above) incorporate mechanisms for handling competition among tasks for attention, and allocation of attention is an emergent property of the architecture without any need for an executive process or special mechanisms.

Model development involved understanding task strategies and priorities used by subjects and encoding that into the model. Understanding the typical multi-tasking management strategies drew upon two major sources. The first one was the tuning subject questionnaire data. In order to enhance and augment the data that we were given, we performed additional cognitive task analysis (CTA) with in-house test subjects. We selected subjects using the same criteria used by BBN for screening subjects. One of the requirements resulting from the initial analysis was to build a model for each of the two aiding conditions—one for text (unaided) and one for the color (aided) condition. Each of these AMBR models represents a generic or average individual. Another key requirement identified during the analysis was that task load should drive both the overall task execution strategy as well as error inducement, under both aiding conditions.

In determining the appropriate level of granularity for implementing specific types of model elements (e.g., micromodels), it was important to understand the contribution , which these elements made to the richness of the human performance aspects of the model. The CTA revealed that there was no clear case to be made for representing action time charges at a low level of granularity. In deciding how to scope and scale model elements, it was necessary to understand the impact of low-level details of the model (e.g., micromodel time charges for hand-movements) and compare them to high-level details of the model (e.g., strategy selection as part of multi-tasking management). As such, it was decided that micromodel calculations would be done at the higher-level for such things as hand-movements and eye actions. In order to be able to tune individual micromodels for calculating action time charges, it was important to be able to control these micromodels using a modifiable coefficient.

## 3.2 Model characteristics

### 3.2.1 Model structure and conceptual overview

As described in Section 2.4 above, the primary components of a COGNET expertise model are a set of cognitive tasks encoding procedural knowledge and a blackboard encoding declarative knowledge about the task situation. A shell allows interaction with the ATC simulator — perceptual information is posted to the blackboard from the simulator and simulator actions are initiated by the model. Figure 3-1 shows a conceptual overview of the AMBR COGNET model.

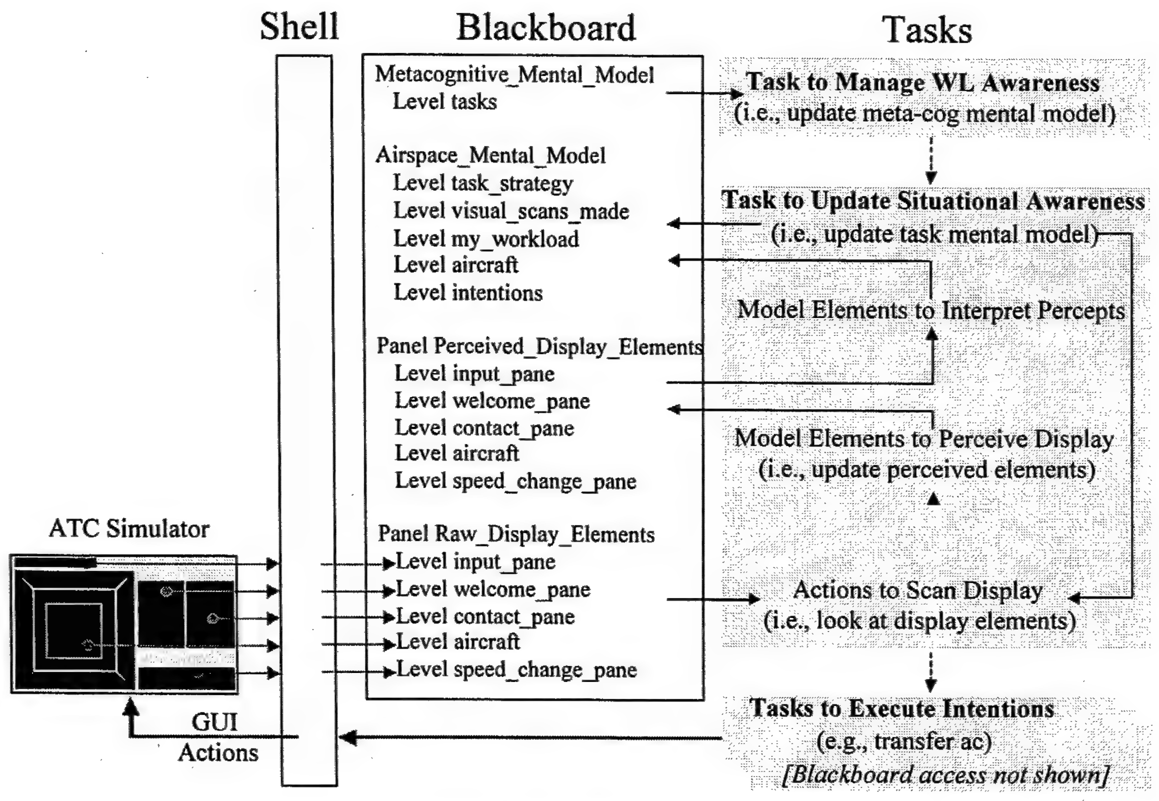

Shell
Blackboard
Tasks
Metacognitive_Mental_Model
Level tasks
Airspace_Mental_Model
Level task_strategy
Level visual_scans_made
Level my_workload
Level aircraft
Level intentions
Panel Perceived_Display_Elements
Level input_pane
Level welcome_pane
Level contact_pane
Level aircraft
Level speed_change_pane
Panel Raw_Display_Elements
Level input_pane
Level welcome_pane
Level contact_pane
Level aircraft
Level speed_change_pane
Task to Manage WL Awareness
(i.e., update meta-cog mental model)
Task to Update Situational Awareness
(i.e., update task mental model)
Model Elements to Interpret Percepts
Model Elements to Perceive Display
(i.e., update perceived elements)
Actions to Scan Display
(i.e., look at display elements)
Tasks to Execute Intentions
(e.g., transfer ac)
[Blackboard access not shown]
ATC Simulator
GUI
Actions

**Figure 3-1. AMBR Model Conceptual Overview**

The model incorporates the notion of 'mental model' by defining an 'Airspace_Mental_Model' blackboard panel. The levels in this panel provide a symbolic representation of the subject's understanding of the task situation. Other panels in the blackboard represent the perceptual process of abstracting from all perceivable display elements (Panel Raw_Display_Elements) those that are perceived as a result of visual scanning of the display (Panel Perceived_Display_Elements).

As part of the analysis and the model building process, it became clear that there needed to be a distinction between 'strategic' and 'interaction' tasks. Loosely, strategic tasks are those that involve understanding the situation and developing intentions to act, while interaction tasks perform each of the required actions. "Update Situational Awareness," which represents the strategies for scanning the display regions and determining what needs to be done, is an example of a strategic task. "Accept Incoming Aircraft," which performs the procedures associated with accepting an inbound aircraft, is an example of an interaction task.

Figure 3-1 shows an overall conceptual view of the ATC model. As events occur in the ATC simulation, the model receives perceptual inputs for these events via the shell through Demons. For example, as text items are added to text lists, as tracks turn different colors, and so on, these events are posted onto their respective levels of the blackboard panel "Raw_Display_Elements." The task "Update Situational Awareness" determines which elements of the radar display should be examined based on the mental model status. As actions are taken to look at various parts of the display while looking for "things to do", aspects of the raw display elements are perceived and posted to the "Perceived Display Elements" blackboard panel. Further, additional model elements in the "Update Situational Awareness" task interpret these percepts and update the mental model accordingly, posting changes to the "Airspace_Mental_Model blackboard panel. Ultimately, it is the posting of intentions to act, to the mental model, that trigger the interaction tasks to initiate and carry out the required steps for the associated action (e.g., Accept Incoming Aircraft).

### 3.2.2 Multi-tasking strategy and attention management

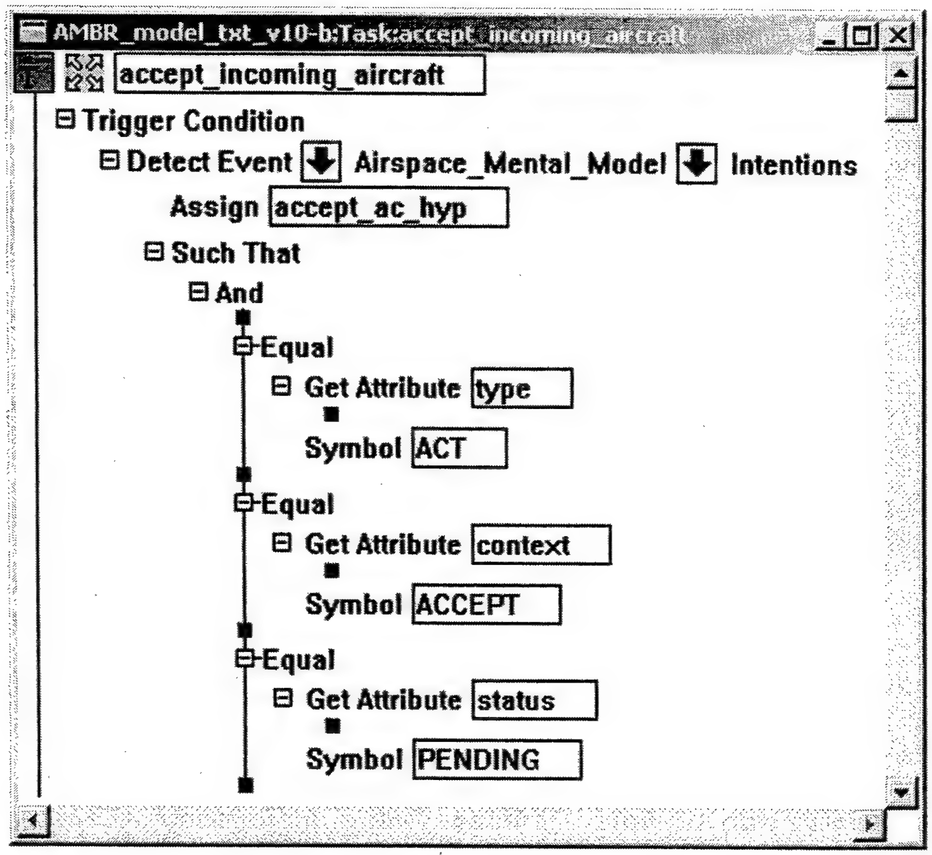
The ATC multi-tasking strategy is accomplished in the model by means of the attention management mechanisms incorporated in the COGNET cognitive architecture. How task selection occurs across multiple tasks, when tasks are or are not interrupted, which tasks are dropped when workload is high, and related implementation choices, is explained below in the context of the cognitive architecture and the reasoning process that is encoded in the ATC model components.

As described in Section 2 above, COGNET tasks are separate chunks of procedural knowledge that compete with each other for attention. Each task has a triggering condition and a priority that are reevaluated anytime there is a change to the blackboard whether initiated by a perceptual demon or a cognitive operation. The triggering conditions indicate when a task is relevant to perform, and evaluation of the priorities of all relevant tasks yields focus of attention to the task with the highest priority.

There are two types of tasks represented in the model — strategic tasks that maintain the mental model and interaction tasks that perform ATC task actions (such as welcoming). Strategic tasks figure out "what to do" then subsequently post intentions to act (pending required actions) to the mental model. Interaction tasks are triggered by these intentions. Interaction tasks in the ATC model are triggered using the "Detect Event" operator. This operator is used to define the context under which a task can be triggered and allowed to compete for attention. Figure 3-2 depicts an example of a typical interaction-level task trigger, specifically, the task trigger for accepting an incoming aircraft. This specific detect event is true if an intention posted to the Airspace Mental Model Panel, Intentions level has a type set to "ACT", a context set to "ACCEPT", and a status set to "PENDING".

The CTA revealed a distinction between strategic and interaction tasks. Subjects consistently indicated that they wouldn't allow themselves to be interrupted when performing an interaction task; once they decided what they needed to do, they would perform the required actions ballistically. High-level strategic tasks could be interrupted and, at a conceptual level, this represents a self-awareness of the need to shift between high and low workload conditions.

COGNET asks are not by definition arbitrarily interruptible and can only be interrupted wherever there is a modification to the blackboard (i.e., POST, TRANSFORM, UNPOST) that is specifically flagged with the "SIGNIFICANT" sub-operator. In order to generate realistic behaviors consistent with the observation that subjects wouldn't allow themselves to be interrupted during an interaction task (e.g., accepting an incoming AC), use of the SIGNIFICANT sub-operator was limited to strategic tasks and was not used in any interaction task.



```
AMBR_model_txt_v10-b:Task:accept_incoming_aircraft
accept_incoming_aircraft
⊟ Trigger Condition
    ⊟ Detect Event  Airspace_Mental_Model  Intentions
        Assign accept_ac_hyp
        ⊟ Such That
            ⊟ And
                ⊟ Equal
                    ⊟ Get Attribute type
                      Symbol ACT
                ⊟ Equal
                    ⊟ Get Attribute context
                      Symbol ACCEPT
                ⊟ Equal
                    ⊟ Get Attribute status
                      Symbol PENDING
```

**Figure 3-2. Task Trigger Using Detect Event**

Two facets of the model's multi-task strategy are the shedding of lower priority tasks during times of high task-load and the shedding of lower priority visual scanning behaviors. The model performs its visual scanning behaviors, by component, looking for something to do. When and if it finds something to do, it "bails out" and attends, at the procedural level, to the task that needs to be performed. If it is aware that multiple actions currently require attention, it will prioritize them based on their relative importance (e.g., the model would attend to accepting a track and then respond to a pending speed change request). The model uses the hierarchy of the goal structure in the 'Update Situational Awareness' task to represent the implicit relative importance of various types of behaviors.

3.2.3 Visual scanning

The visual scanning heuristics represented in the model are based on analysis of test subject data conducted as part of the cognitive task analysis (CTA). Two simplifying assumptions that were integrated into these representations are that 1) changes in the display could be consistently perceived (but not the quantity nor type of change) and that 2) the mental model that drives the scanning heuristics wasn't capable of "forgetting". Although the model structure and iGEN™ application both support the ability to apply memory-moderators, since there were no empirical data to support a principled approach to their construction and use for this specific task, they were not applied.

The typical scanning strategy represented in the model is to scan the display, region by region, for the visual indicators of "something to do". Those display regions that haven't changed from the last visual scan are ignored. This status is part of the mental model. For text list panes, the model scans the list and compares what it sees to its mental model of what it has read in order to distinguish things it has done from things it still needs to do. The scanning heuristic represented in the "text" model is purposefully different from the scanning heuristic represented in the "color" model. While the text model has goals mapped to scanning each of the various ATC display components (e.g., radar display, accept text pane, speed change text pane), the color model has the singular goal of scanning the radar display for color changes. This difference between the two models is consistent with the results of the CTA that revealed two fundamentally different approaches for visually scanning for "something to do" for the text and color conditions.

### 3.2.4 Blackboard

There are two blackboards in the ATC model; the standard blackboard and the metacognitive blackboard. As is typical of iGEN™ models, these two blackboards play a central role in the design and execution of the AMBR ATC cognitive model. Figure 3-3 below depicts the high-level blackboard structure for the standard blackboard and Figure 3-4 that for the metacognitive blackboard. Each panel, and its subordinate levels, is also enumerated. The ATC model standard blackboard contains four primary types of knowledge and information such as declarative knowledge concerning task concepts, raw perceptual inputs from the external world, percepts of these raw inputs, and a mental model for encoding symbolic representations of dynamic context. The standard blackboard is broken out into the following Panels:

- Raw ATC Display: As events that can be perceived occur in the simulation environment, they are sent to the model through a Demon and posted to the appropriate level of this panel based on the type of perceptual information that is encoded.
- Perceived Display Elements: As the model invokes visual scanning actions, it posts representations of the resulting percepts onto the appropriate level of this panel
- Airspace Mental Model: This panel stores such things as action status, introspective task load assessment and intentions. These intentions connect the model's volition to act and the action instantiation in the simulation console.
- ATC Task Concepts: This panel provides the knowledge representation for key ATC declarative knowledge components such as ATC controller, AC color, and display region semantics.

The metacognitive blackboard is broken out into the following Panels:

- Metacognition: Moment-to-moment task state information for each active task is stored in this panel. The information includes task start time, task [trigger] context, interruption and resumption information, task priority, as well as workload metrics.

- Self: This panel stores information that is used for introspection and self-awareness of ATC task. This includes information for use in task load determination, a representation of expertise level, and actions taken by the model in the situated environment.

### 3.2.5 Cognitive tasks

As discussed previously, the model distinguishes between strategic and interaction tasks. Strategic tasks figure out "what to do" then subsequently post intentions of pending required actions to the mental model. Interaction tasks are triggered by these intentions and contain the knowledge required to execute the procedure-specific actions. Table 3-1 lists the strategic and interaction tasks in the ATC model. A single example of each of these task types is presented in more detail in the paragraphs that follow.

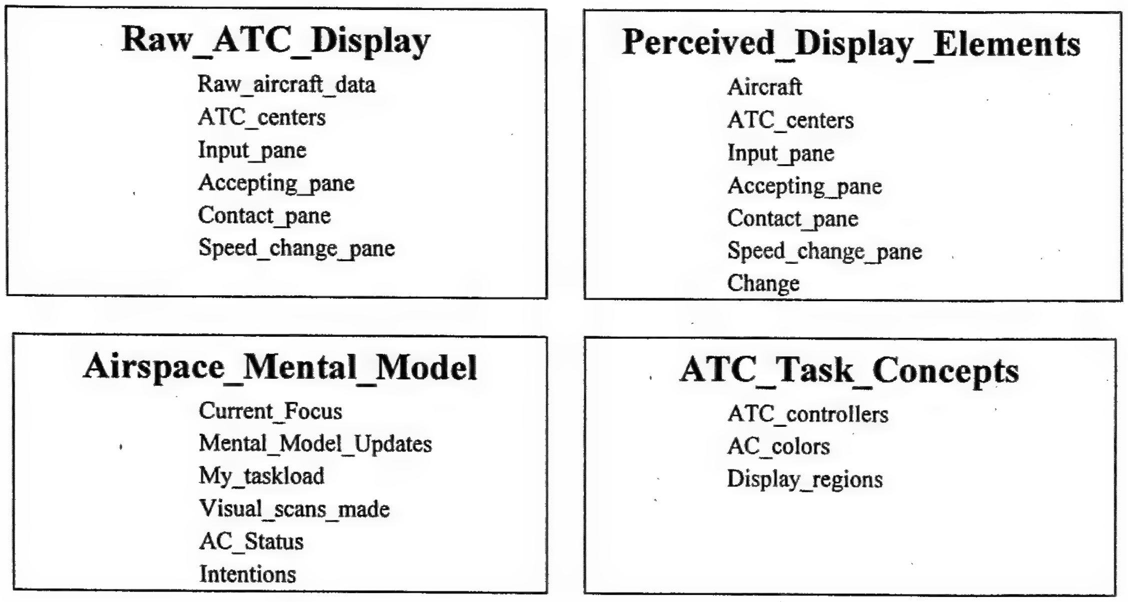


**Figure 3-3. ATC-Model Standard Blackboard Structure**

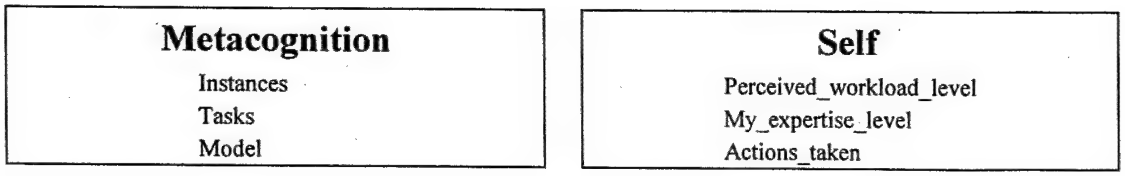


**Figure 3-4. ATC-Model Metacognitive Blackboard Structure**

**Table 3-1. ATC Model Task Listing**

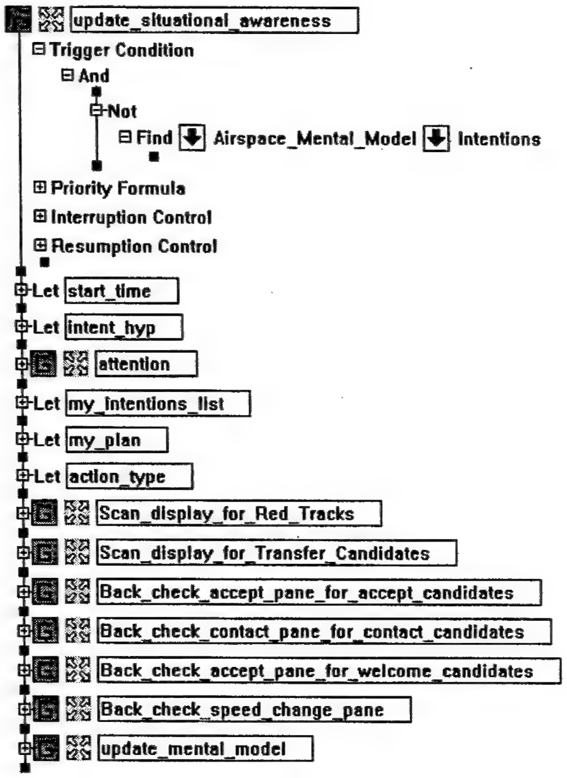
| Task Name | Type | Description |
|---|---|---|
| Think_about_task_load | Strategic | Maintains self-awareness of changes in task load |
| Update_situational_awareness | Strategic | Maintains situational awareness of ATC task elements |
| Accept_incoming_aircraft | Interaction | Accepts incoming AC |
| Welcome_incoming_aircraft | Interaction | Welcomes incoming AC |
| Respond_to_speed_increase_request | Interaction | Responds to speed change requests |
| Transfer_outgoing_candidate | Interaction | Transfers outgoing ac |
| Order_ac_to_contact_ATC | Interaction | Orders ac to contact ATC |
| Get_aircraft_out_of_holding | Interaction | Responds to red tracks to get them out of holding |

The ATC model task "Update situational awareness" (SA) represents the central task of visually inspecting the radar display for "something to do". It maintains the ATC model's mental model of the dynamic situation – information necessary to decide what to do, when, and how. Among other properties, the mental model contains the status of any given aircraft attended to by the model and notions of task status for each respective aircraft that has been handled.

The SA task represents the scanning precedence of display regions implicitly through its goal structure. Figure 3-5 shows the high-level view of the SA task. The visual scan goal list for the SA task is constructed so that each goal is tested, sequentially, for a given task execution. Part of the precondition for each goal is a test for the creation of intentions that were created during the execution of preceding goals. If any goal that is executed leads to one or more intentions being created, the remaining display regions are not scanned during that instance of task execution.

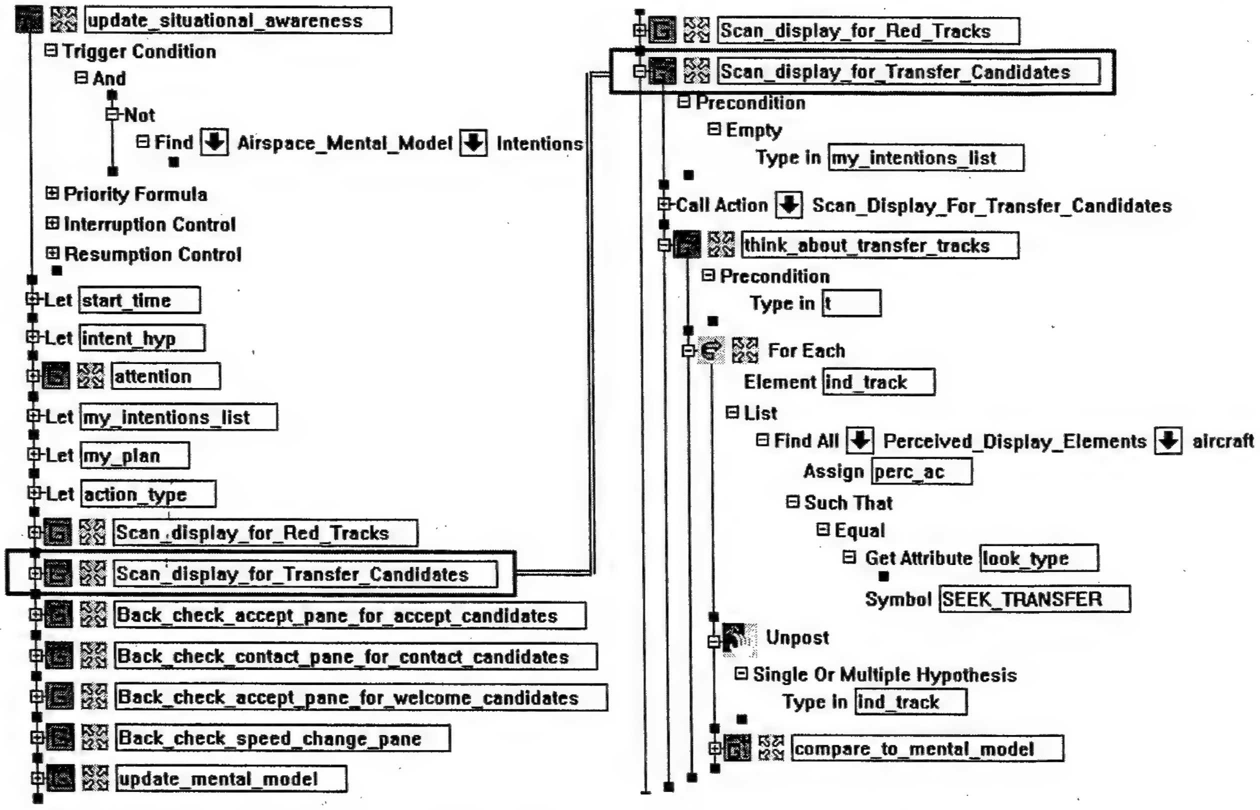
The goals are executed in sequence until the intentions level of the Airspace mental model panel of the blackboard has an action intention. Each triggering of the SA task results in either something to do (one or more intentions posted to the mental model) or nothing to do (no intentions posted). A more detailed example of part of the Update Situation Awareness Task (the scan for transfer candidates) is shown in Figure 3-6.

The interaction tasks carry out the action intentions posted in the mental model. An example of an interaction task is shown in Figure 3-7. This task, like the other interaction tasks, is composed primarily of a sequence of visual scanning and manual manipulation actions.



```
update_situational_awareness
  ⊟ Trigger Condition
      ⊟ And
          Not
              ⊟ Find  Airspace_Mental_Model  Intentions
  ⊞ Priority Formula
  ⊞ Interruption Control
  ⊞ Resumption Control
Let start_time
Let intent_hyp
attention
Let my_intentions_list
Let my_plan
Let action_type
Scan_display_for_Red_Tracks
Scan_display_for_Transfer_Candidates
Back_check_accept_pane_for_accept_candidates
Back_check_contact_pane_for_contact_candidates
Back_check_accept_pane_for_welcome_candidates
Back_check_speed_change_pane
update_mental_model
```

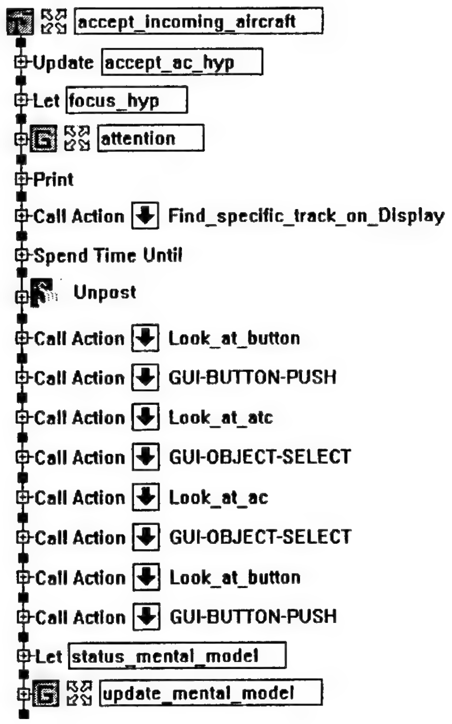
**Figure 3-5. High-level View of SA Task**



```
update_situational_awareness
  ⊟ Trigger Condition
      ⊟ And
          Not
              ⊟ Find  Airspace_Mental_Model  Intentions
  ⊞ Priority Formula
  ⊞ Interruption Control
  ⊞ Resumption Control
Let start_time
Let intent_hyp
attention
Let my_intentions_list
Let my_plan
Let action_type
Scan_display_for_Red_Tracks
Scan_display_for_Transfer_Candidates
Back_check_accept_pane_for_accept_candidates
Back_check_contact_pane_for_contact_candidates
Back_check_accept_pane_for_welcome_candidates
Back_check_speed_change_pane
update_mental_model
```

```
Scan_display_for_Red_Tracks
Scan_display_for_Transfer_Candidates
  ⊟ Precondition
      ⊟ Empty
          Type in my_intentions_list
  Call Action  Scan_Display_For_Transfer_Candidates
  think_about_transfer_tracks
    ⊟ Precondition
        Type in t
    For Each
        Element ind_track
      ⊟ List
          ⊟ Find All  Perceived_Display_Elements  aircraft
              Assign perc_ac
            ⊟ Such That
                ⊟ Equal
                    ⊟ Get Attribute look_type
                        Symbol SEEK_TRANSFER
      Unpost
        ⊟ Single Or Multiple Hypothesis
            Type in ind_track
      compare_to_mental_model
```

**Figure 3-6. Update Situation Awareness Task Extract**



```
accept_incoming_aircraft
Update accept_ac_hyp
Let focus_hyp
attention
Print
Call Action Find_specific_track_on_Display
Spend Time Until
Unpost
Call Action Look_at_button
Call Action GUI-BUTTON-PUSH
Call Action Look_at_atc
Call Action GUI-OBJECT-SELECT
Call Action Look_at_ac
Call Action GUI-OBJECT-SELECT
Call Action Look_at_button
Call Action GUI-BUTTON-PUSH
Let status_mental_model
update_mental_model
```

**Figure 3-7. Interaction Task Example — Accept Incoming Aircraft**

### 3.2.6 Actions and action time charges

The ATC model is capable of two types of actions; those that represent visual scanning behaviors (such as "seek transfer candidate") and those that represent motor actions for interacting with the ATC simulator console (such as "press accept button"). As previously mentioned, each of the respective interaction tasks encodes the motor-level actions (iGEN™ perform-action operations) required to perform the corresponding procedure. Table 3-2 below describes each of the actions that the ATC model is capable of performing.

In order to generate realistic estimates of overall reaction times, it is necessary to encode metrics for calculating predicted action time charges. These are represented inside micromodels in the ATC model. iGEN™ micromodels are granularity-neutral and don't require any pre-determined level of representation. The selected level of representation allowed the model to use the quantity and location of display elements to calculate action time charges. This approach of "just enough representation" was motivated by the ever-present tradeoff between choosing a granularity level low enough to generate high-fidelity human performance predictions and providing a feature-rich model that captures as much of the full task domain as possible given the development resources available.[4]

[4] It was precisely to support this kind of tradeoff that COGNET and CGF-COGNET were constructed with the flexible granularity stance described in 2.1.1.

Table 3-2. **ATC Model Action Time Charges in Micro Models**

| ACTION | MICROMODEL (ms) |
|---|---|
| **GUI-OBJECT-SELECT** | Fixed charge given dynamic taskload level high 580, med 600, low 650 |
| **GUI-BUTTON-PUSH** | Fixed charge given dynamic taskload level high 580, med 600, low 650 |
| **Back_check_accept_pane_for_accept_candidates** | 200 * number of items in accept pane |
| **Back_check_accept_pane_for_welcome_candidates** | 200 * number of items in accept pane |
| **Back_check_speed_change_pane** | 300 * number of items in speed change request pane |
| **Scan_Display_for_any_Red_Tracks** | 300 * number of red tracks on display |
| **Scan_Display_For_Transfer_Candidates** | 200 * the number of perimeters that are searched until candidate found (4 max) |
| **Backcheck_Contact_Pane_for_Contact_candidates** | 300 * number of items in contact pane |
| **Look_at_pane** | Fixed charge given dynamic taskload level high 580, med 600, low 650 |
| **Look_at_button** | Fixed charge given dynamic taskload level high 580, med 600, low 650 |
| **Look_at_ac** | Fixed charge given dynamic taskload level high 580, med 600, low 650 |
| **Look_at_atc** | Fixed charge given dynamic taskload level high 580, med 600, low 650 |
| **Look_for_collision_conflicts** | 300-500, depending on context |

### 3.2.7 Errors

As in any attempt to generate realistic human performance data, the construction of the ATC model required the modeling of performance errors. Any effort to induce error in a human performance model needs to take into account the relationship between the various types of errors possible, the underlying human information processing mechanisms responsible for those errors, and the error recovery processes necessary to correct those errors. The CTA revealed that at least four different types of errors were possible, as outlined in Table 3-3 below. It was clear that the impact and interaction of various types of errors, and the necessary error-recovery strategies, were complex and rich. Unfortunately, there was no available empirical data to clarify the possibly complex and cascading relationships between these error types and the observed performance measures (e.g., an instance of an aircraft symbol becoming red). As a result, there was no clear way to understand how each type of error might have contributed to the performance results of the human subjects, and therefore how to model the underlying errors. As the time taken to handle/correct errors can easily alter the response time statistics, the assessment of fit between human and model performance measure results is a complex issue. If the error results map well but the response time results do not, their lack of independence suggests that the overall mapping is poor, and so on.

In the absence of a clear understanding of the errors captured by the fly-off performance measures, it was decided that the scope of errors that would be integrated into the model would

**Table 3-3. Error Types in ATC Task Performance**

| Type | Example |
|---|---|
| Cognitive | Mis-process an AC name |
| Mental Model | Mis-remember the status of an AC and repeat an action |
| Perceptual | Mis-perceive an AC name and perform action on incorrect track |
| Motor | Press the wrong button |

be those that did not require significant amounts of error-recovery logic. After analyzing the subject data, it was decided to generate errors at the motor level by assigning a probability to the inducement of such errors and stochastically inducing this error type based on the dynamic task load. Specifically, when the error probability was met for a given AC that was required to be transferred, the model would incorrectly press the "contact" button instead of the "transfer" button. The result is that the track eventually turns red because of the incorrect actions being applied. The components of the model that respond to a track turning red are used to recover from this error (i.e., task "get_ac_out_of_holding"). Moreover, this error inducement was capable of effecting multiple performance score components, such as total score, holding delays, duplicated messages, and incorrect messages.

## 3.3 Model development process

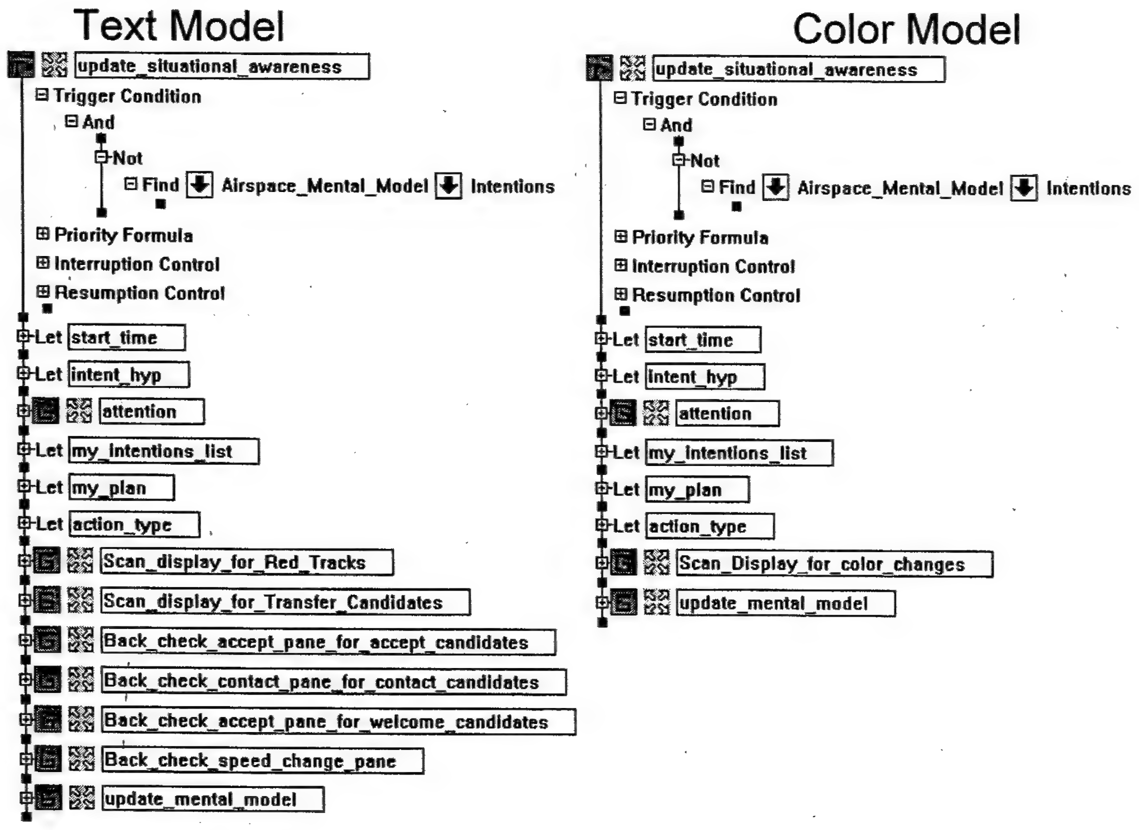
The model development process unfolded over three primary stages. The first was the development of the initial baseline competence model encompassing the knowledge required to do the more-complex job (the text condition). The second stage involved refinement of the competence model into a realistic performance model yielding human-like response time and accuracy behavior. Once the model produced good performance and response time measures, the model was split into two versions — one for the text condition and one for the color condition. Each stage will be discussed in turn.

The initial baseline model was a competence model in that it contained all the knowledge needed to perform the job. However it was characterized by the fact that at this early stage there were no errors, the visual search heuristic was exhaustive, and each individual task was performed properly, but without a realistic temporal priority scheme. This competence model, while representing the baseline knowledge required to "do the job" did not perform the tasks in a realistic order, nor did it generate realistic response times.

Refinement of the model into a performance model involved fine-tuning the task strategy, tuning the micromodels using the human subject tuning data, and adding error inducement. First, the visual search strategy was modified from an exhaustive search through all display regions to a search until first match. As discussed above, the final models incorporated a visual scan strategy that searched regions in order of importance and when something to do was found, stopped the search and performed the action. This change produced a more realistic task performance ordering in that it more closely matched the strategy employed by test subjects.

The second component of performance involved refinement of response times generated by the model. Model response time data were compared to tuning subject response times and the basic operation times in the micromodels were adjusted up or down to tune the micromodels. This was repeated until the model produced response times in the range of the times produced by the tuning subjects. The third aspect of model refinement involved error inducement. The method for incorporating errors into the model was described in Section 3.2.7 above. It was at this second stage in model development that the generation of subjective workload assessment measures was added to the model. This is described below in Section 3.4.

In the final stage of model development , separate model versions were created for the text and color conditions. Throughout the model development, the focus had been on creating a model that could handle the more complex text condition, with the rationale that the color condition was a simpler derivative model. Creating the color model involved changing only the "Update_situation_awareness" task, since that is the task that figures out what to do. The interaction tasks did not change. As the CTA indicated that subjects in the color condition ignore the text displays and focus solely on the color changes, there was only one visual scanning goal in the color model substituting for the sequence of display region scans in the text model. This difference is shown in Figure 3-8.



```
Text Model
update_situational_awareness
  ⊟ Trigger Condition
    ⊟ And
      ⊟ Not
        ⊟ Find  Airspace_Mental_Model  Intentions
  ⊞ Priority Formula
  ⊞ Interruption Control
  ⊞ Resumption Control
⊞ Let start_time
⊞ Let intent_hyp
⊞ attention
⊞ Let my_intentions_list
⊞ Let my_plan
⊞ Let action_type
⊞ Scan_display_for_Red_Tracks
⊞ Scan_display_for_Transfer_Candidates
⊞ Back_check_accept_pane_for_accept_candidates
⊞ Back_check_contact_pane_for_contact_candidates
⊞ Back_check_accept_pane_for_welcome_candidates
⊞ Back_check_speed_change_pane
⊞ update_mental_model
```

```
Color Model
update_situational_awareness
  ⊟ Trigger Condition
    ⊟ And
      ⊟ Not
        ⊟ Find  Airspace_Mental_Model  Intentions
  ⊞ Priority Formula
  ⊞ Interruption Control
  ⊞ Resumption Control
⊞ Let start_time
⊞ Let intent_hyp
⊞ attention
⊞ Let my_intentions_list
⊞ Let my_plan
⊞ Let action_type
⊞ Scan_Display_for_color_changes
⊞ update_mental_model
```

**Figure 3-8. Comparison of Update Situation Awareness Task for Text and Color Models**

## 3.4 Subjective workload assessment

In addition to performing the air traffic control task in a human-like manner (i.e., that fits control data from human subjects performing the same task), the AMBR model was also required to provide assessments of various subjective aspects of task performance centering on the construct of subjective workload. Before discussing how this was accomplished, we offer some thoughts on the concept of workload, which has been used in a variety of different ways in the AMBR program as well as in the broader literature.

Workload is sometimes used to describe an input to the task, (e.g., high-workload versus low-workload scenarios), sometimes as an internal state in the human (or model) that can affect task performance (e.g., workload-based strategies), and sometimes as an output of the process (e.g., ratings products by the model/person after task completion). This has contributed to some conceptual confusion about the denotata of the term. From our perspective, workload is not a state or a component, but is rather a relationship between an information processing system and its (complex) environment. As noted earlier in the central COGNET 'equations', performance arises from interaction between the problem/environmental dynamics, and an information processing system, which contains both internal processing mechanisms and internal expertise. Each of these contributes to the relationship that can be called workload.

For example, the expertise within the information processing system may be more or less useful in achieving the system's goals, given the current state and dynamics of the external environment. When the expertise is less useful, more internal processing may be needed to achieve the goal, or only less-than-goal conditions may be achievable. Similarly, the pace/complexity of the events in the environment may require more internal processing by the system to achieve/maintain its goals, independent of the usefulness or efficiency of the underlying knowledge involved. In any case, the amount of momentary internal processing is driven by this relationship between the information processing system's goals, expertise, and processing mechanisms and the current state and dynamics of its internal environment. This viewpoint helps differentiate the three varying notions of workload noted above, and defines an approach to the workload subjective self-assessment task.

A given external condition may involve much less processing for an individual with more expertise than one with less. Thus, workload cannot be, by itself, an input to the task. Similarly, there is no internal register (at least within the COGNET framework), which reflects the momentary status of the internal processing system's workload and adjusts it accordingly. Rather, the ability of people to adjust their strategy to the dynamics of the internal processing mechanisms varies from person to person, and is arguably learned. This makes it a matter of expertise not mechanism, and this kind of expertise is clearly metacognitive, as discussed above (in 2.4). However, the ability to modulate the usage of expertise according to the status of the internal information processing system's dynamics requires some access to this information. That is, it requires a self-awareness of the status and dynamics of the internal processing mechanisms. This is also a metacognitive function, as discussed in 2.4. Workload-based strategy adjustment can thus be interpreted as a cognitive process based on metacognitive knowledge specific to the AMBR task domain, and dependent on declarative self-awareness of the internal processing dynamics. This aspect of the process was discussed above (3.2). Workload self-assessment, then, is a process of introspectively (and retrospectively) reporting on this self-awareness information.

The overall approach to generating these introspective retrospective accounts involved four basic steps. In the first step, the six separate measures quantified by the human subjects were analyzed and conceptually mapped onto the internal architecture and the Principles of Operation of the COGNET system. This conceptual mapping was taken further in the next step, as each of the six measures was then operationalized in terms of specific aspects of metacognitive self-awareness, whether they existed within BATON or not. Those that did not already exists were then implemented within BATON (in a third step), and combined with those that had previously been implemented to allow the self-reporting of the subjective workload assessments required. This resulted in the ability of the model to introspect and create a post-hoc report of its own workload on the six measures involved. The scale of those measures was not necessarily the same as that used by the measurement instrument, i.e., the quasi-interval scales used by the human subjects. In a fourth and final step, the measures generated by the model were calibrated to those specific measurement scales. Details of each step are defined below.

### 3.4.1 Relating TLX Measures to COGNET Constructs

The AMBR moderator had the human subjects provide subjective assessments of six characteristics of the overall task, based on TLX, NASA's Task Load Index (Hart & Staveland, 1988). These were:

- Physical demand – defined as "how physically demanding was the task"
- Mental demand – defined as "how mentally demanding was the task"
- Temporal demand – defined as "how hurried or rushed was the pace of the task"
- Performance self-estimate – defined as "how successful were you in accomplishing what you were asked to do"
- Effort estimate – defined as "how hard did you have to work to accomplish your level of performance"
- Frustration – defined as "how insecure, discouraged, irritated, and annoyed were you"

Each subject provided an estimate of each characteristic on a Likert-like scale divided into 10 intervals, and measured as 21 discrete values (integers 0-10 inclusive plus all intermediate half-values). Thus, the model had to produce estimates of these same variables on this same scale. The general theoretical framework for this process was one of relating each measure to the self-awareness of various aspects of the processing within the COGNET representation, as described above. The analysis of each measure in terms of COGNET constructs is given below:

**Physical demand**. This construct was based on self-awareness of the overall level of activity of the motor system. The AMBR model did not need to declare and monitor individual motor resources (e.g., separate hands, eye resources, etc.), but rather directed action to the motor system on these channels using different Perform Action operations. Each action duration was, in addition, given a duration by a specific micromodel. The awareness of physical workload is thus accomplished in two stages. First, the system had to be aware of (collect) the total time spent in each kind of action (e.g., move mouse, move eyes, give verbal message). Second, weights had to be assigned to discriminate low effort actions (e.g. eye-movements) from high-effort actions. The physical workload value could then be calculated as the weighted sum of the action times across all actions, normalized to the length of the scenario.

**Mental demand**. This construct was based on the self-awareness of the cognitive complexity of the various tasks being executed and their relative frequency. Mental workload was estimated separately for each cognitive task in the model. Because tasks can be carried out in many different ways, the average *complexity* of each task during the execution of a scenario is calculated as the average number of goals and method calls during task execution (including the number of goals and method calls within the higher level methods called). The task complexity was then multiplied by the number of times the task was performed in the scenario. This weighted complexity was summed across all tasks, and normalized to the length of the scenario.

**Temporal demand**. This is a difficult construct to relate to COGNET internal structure and dynamics. It was ultimately decided to do so in terms of a complementary construct, the sense of idleness or momentary sense of 'nothing to do now.' It was felt that the more frequently the model had such an awareness, the less time-pressure it would feel, and vice versa. This sense would occur, in model terms, when the model was aware that it had no 'queued' tasks, was scanning the screen for new tracks needing attention, and finding none. Thus, a self-awareness capability for this 'nothing to do' state was needed, and its negative value used to measure temporal workload, normalized to the length of the scenario. This use of negative value is analogous to the well-known use of 'regret' to measure (negative) utility in decision theory.

**Performance**. A similar regret-like approach was selected to relate this measure to COGNET constructs. The model does not have any awareness of how well it is doing or how it is performing on a positive side, but it does have an awareness of when it makes errors of omission or commission, because it attempts to correct those error when it becomes aware of them. Thus, the more errors the model is aware of, the worse its perceived performance would be, and when it is aware of no errors, its self-perception of its performance would be very high (whether it would be so empirically or not). The performance measure was ultimately assigned as the negative value of the count of perceived errors during a scenario, normalized to the length of the scenario.

**Effort**. Conceptually, the concept of effort as defined for the human subjects seems to refer to the total amount of time they spent 'working' in the scenario, possibly excluding or discounting the visual effort of scanning the display. (Why discount visual scanning? Although there is no data to confirm or disconfirm this position, it seems likely that a scenario in which no tracks appeared would be said to have involved 'no effort', even though the eyes would still scan the screen, at least periodically). In the CGF-COGNET variant of the BATON architecture, there can be multiple threads of work going on in parallel at any time, for example hands, voice, and eyes, all with different and potentially overlapping action start and end times. Because of this parallelism, it would be incorrect to simply sum all the work times, as many would/could be parallel. Alternatively, the total time in which some motor action were occurring could be collected as the total amount of 'busy' time in the problem. Thus, the reporting of ‘effort’ required an awareness of the states when some activity was occurring on any motor channel. When divided by the total time of the scenario, it would yield a measure of the proportion of time spent working the problem, and thus was used to report the effort involved in the scenario.

**Frustration**. This is an extremely ill-defined concept, even to people (as evidence by the range of emotional states that subjects were told to map into the term 'frustration, and by the generally low inter-subject correlations on this scale). After considering a number of aspects of the model and execution architecture, that could be related to one or more of the terms used in the elicitation frame (i.e., insecurity, discouragement, irritation, annoyance), it seemed that a simple approach was best. The approach selected is based on an awareness of the times that a cognitive task is interrupted by another cognitive task. While the nature of the testbed work is such that interruption is minimized (i.e., because it's HCI provides limited ability to leave work elements in an interrupted state), there are still opportunities for this to occur, primarily with visual and/or purely interpretative tasks. Thus, frustration was simply reported as the number of cognitive task interruptions divided by the length of the scenario. It is noted, however, that human subjects overall found the text-based version of the interface more frustrating than the color version, although it is not clear in terms of the model why this was the case. In-house data collected from human subjects at CHI Systems further showed that there was a strong sequence effect, in that subjects were much more frustrated with the text version *after having first used the color version.* However, since the design did not allow any knowledge of prior runs to be incorporated into this self-assessment process, this effect could not be used in producing the model's frustration self-report. Thus, it was expected that the form of self-reporting selected for this measure would under-report frustration for the text-only case, an expectation that was borne out by the empirical data (see below).

### 3.4.2 Operationalization of the Measures in BATON Terms

The preceding analysis translated the TLX workload constructs into self-awareness terms consistent with our underlying representation of subjective workload reporting. This analysis was next used to develop specific computational operationalizations of the measures within the BATON cognitive architecture. In this operationalization process, effort was made to (re-)use various aspects of self-awareness that had already been implemented within:

- the CGF-COGNET version of the BATON software used to build the AMBR model, and
- other variants of BATON, particularly the ORGNET variant being created to support analysis and allocation of cognitive functions across multiple human and/or computational agents in a complex task environment (see Weiland and Eilbert, 2000; Weiland, Zachary, and Le Mentec, 1998; Zachary, Weiland, and Le Mentec, 1997)

When this was not possible, new CEL operators were defined as needed to provide explicit self-awareness of specific model states (such as awareness of having made an error. In some cases, as noted below, it was also necessary to define additional instrumentation of BATON to create specific types of self-awareness of physical actions (either motor actions or deliberative perceptual actions).

The specific formula developed for producing each individual workload measure is provided in Table 3-4. The table uses the following elements of notation:

- The symbol TS in all cases refers to the length (in time) of the scenario.
- The functions that are in capital letters are self-awareness measures taken from the ORGNET version of BATON.
- The variables in italics represent new CEL operators that will be used to measure individual events, such as awareness that an error has occurred, or new instrumentation inserted into BATON to quantify a specific aspect of the system's execution.

## 3.4.3 Implementation of Measures in BATON

The implementation of the six subjective workload measures defined in Table 3-4 was done using the existing metacognitive functionality in the BATON software, specifically the metacognitive self-awareness blackboard panel. In addition to the instrumentation of the

**Table 3-4. BATON Operationalization of AMBR Workload Measures**

| **Measure** | **Computational Formula** | **Operationalization** |
|---|---|---|
| Physical | $(\sum_{\text{all } i} k_i \bullet \textit{action-time}_i)/T_S$ | Where i is an action type, $k_i$ is the weight of that action type, and *action-time*$_i$ is a function which collects total time spent taking that action during a scenario by instrumenting the action mechanism |
| Mental | $(\sum_{\text{all } i} DPC_i \bullet DTT_i)/T_S$ | Where i is a cognitive task, and $DPC_i$ and $DTT_i$ are ORGNET measures |
| Temporal | $-(\sum_{\text{all } i} \textit{nothing-to-do}_i)/T_S$ | Where *nothing-to-do*$_i$ is a CEL operator that returns a value of one iff the visual scan task has been completed and the model found nothing to do |
| Performance | $-(\sum_{\text{all } i} \textit{error-detected}_i)/T_S$ | Where *error-detected*$_i$ is a CEL operator that returns a value of one iff a cognitive task found that the model had made an error and was about to try to correct it (if correctable) |
| Effort | $(\sum_{\text{all } i} \textit{time-spent-in-work}_i)/T_S$ | Where i is an increment in time, and *time-spent-in-work*$_i$ is a function which measures the time within that increment in which motor activity is occurring on any execution thread by instrumenting the action mechanism |
| Frustration | $(\sum_{\text{all } i} DTI_i)/T_S$ | Where $DTI_i$ is the ORGNET measure of the number of times cognitive task i was interrupted during the scenario |

information processing mechanisms that make up the CGF-COGNET version of BATON, several self-awareness measures were integrated from the ORGNET variant of the system. These were:

- DPC – a measure of the Dynamic Procedural Complexity of a task, it essentially counts the number of GOALs, DETERMINE operations and CALCULATE operations that are actually executed within a given problem-instance (i.e., scenario). For AMBR, the DPC measure was extended to incorporate all the GOALs, DETERMINE operations and CALCULATE operations that were executed as part of METHODS that were invoked during execution of any instance of the task as well.
- DTT – a measure of Dynamic Task frequency, it counts the number of times the task is actually executed in a given problem-instance.
- DTI – a measure of Dynamic Task Interruption, it counts the number of times that a given task has been interrupted, while executing, by other tasks.

The integration of these measures made the information they collect part of the system's self-awareness on the metacognitive blackboard.

Two additional CEL operators also had to be created. These were

- 'error-detected' – this is an operator that, when executed, places an awareness that the model has made an error of some sort onto the metacognitive blackboard;
- 'nothing-to-do' – this is an operator that, when executed, places an awareness onto the metacognitive blackboard that the model has found nothing to do (other than scan).

Creation of CEL operators is a relatively simple process given the design of the BATON software, requiring usually less than an hour to define and recompile into the software. The above are both metacognitive operators that placed an awareness of the named condition on the metacognitive blackboard, based on where it was encountered and executed. Once created, the operators had to be added to the expertise model in the appropriate locations. The new-operator creation process automatically integrates the new operator into the CGR interface, making insertion of the new operators a straightforward 'point and click' operation.

Finally, two new types of instrumentation of the BATON mechanism had to be created. These two new types of cognitive proprioception were:

- 'time spent in work' – an instrumentation of the scheduling mechanism that collects the total amount of time during which time was being consumed (via a Spend-Time operator), on any thread of activity within the system;
- 'action time' – an instrumentation of the motor action mechanism that collects, for a specific action type, the total amount of time during which time was spent performing that action via a Spend-Time operator.

Developing this functionality involved relatively small amounts of effort as well. As discussed in the conclusions, a total of 3% of the project effort was expended in all the modifications to BATON made in the effort. This total includes the integration of the ORGNET measures, creation of the new CEL operators, and creation of the new cognitive proprioception functions, as well as their testing and debugging.

### 3.4.4 Calibrating the Measures to the TLX Scale

It should be clear that the formulae in Table 3-4, while arguably measuring the various self-perceptions collected in AMBR, do not do so on the same Likert-like scale used by the human subjects. In addition, the values generated by the model were not inherently tuned to the relative differences in the judgements made by the human subjects. Thus, the final step in implementing the measures in Table 3-4 was a calibration process that dealt with these two issues.

Each subject in the initial tuning data set provided an assessment on each of the six subjective factors for six different scenarios. For each given factor and scenario i there was then a vector of j subject values $x_{ij}$ on the scale for that factor, and a statistic $X_i$ which represents the mean value of the $x_{ij}$ for scenario j. For each of those same six scenarios, the model was able to produce a value $y_i$ for the same factor, using the relevant measure from Table 3-4. This yielded a set ordered pairs $(X_i, y_i)$, representing the model's assessment and average subject assessment for that factor on scenario i. However, the $y_i$ represent values on an open-ended ratio scale, while the $X_i$ represent interval values on a closed-ended scale. To map the first into the second, a regression was done to calculate the $X_i$ as a function of the $y_i$. This regression process yielded a linear function which then stated how the $y_i$ could be transformed, once calculated, to yield a calibrated measure on the same scale used by the human subjects. If this function is written $F_i(Y)$, then the final value of the model's measure for factor i is defined as:

$F_i(Y)$ if in the interval [0,10],

0 if $F_i(Y) < 0$, and

10 if $F_i(Y) > 10$.

The resulting equations are shown in Table 3-5. In each line of the table, the directly generated measure is named in capitals (e.g., EFFORT), and the calibrated measure is shown with an asterisk (e.g., EFFORT*)

**Table 3-5. Calibration formula for AMBR Workload Measures**

| **Measure** | **Calibration Formula** |
|---|---|
| Physical demand (PHYSICAL) | PHYSICAL* = 9.1949*PHYSICAL + .7955 |
| Mental demand (MENTAL) | MENTAL* = 7.9385*MENTAL+ .5032 |
| Temporal Demand (TEMPORAL) | TEMPORAL* = -7.9865* TEMPORAL + 6.7737 |
| Performance (PERFORM) | PERFORM* = 380.8318*PERFORM + .7606 |
| Effort (EFFORT) | EFFORT* = 4.6704*EFFORT + 1.2269 |
| Frustration (FRUSTRATE) | FRUSTRATE* = .1703*FRUSTRATE - .7498 |

### 3.4.5 Workload Assessment Results

The COGNET/iGEN™ model was the only model to generate subjective self-assessments for all six of the individual workload measures used by the human subjects. As a result, the models were compared at the model comparison 'fly-off' only in terms of a single composite workload self-assessment score. In the case of the COGNET/iGEN™ model, this was obtained simply by averaging the six individual scores to yield an overall workload score.

However, because the individual workload scores were calculated by the COGNET/iGEN™ model, it is possible to compare them individually to the human subject scores as well. The comparisons with 'fly-off' subject data are shown below in Figures 3-9 through 3-14. In general they show a good fit with the human data, increasing from low-load to high-load problems. The model appears to generally view the aided (i.e., color) condition as less demanding than the unaided (i.e., text-only) condition, but less so than the human subjects who often saw even the highest track-load scenario with the color condition as involving less workload than the lowest workload text condition. The expected divergence appeared with the frustration measure, as the model did not find the text condition as frustrating as the human subjects did. This was further complicated by the fact that the model, for some reason, found the intermediate track load problem more frustrating than the high-track-load problem for the unaided condition.

One additional methodological remark is offered here concerning the calibration process. Due to the time schedule of the project, the calibration process was done in only two iterations. That is, an initial calibration was done, and then parameters in the workload prediction parts of the model (e.g. the weights of the action types the physical demand measure) were revised. It is likely that given a few additional iterations a much tighter fit with the human subjects' assessments could have been obtained. At the same time, it is not clear that there would have been much value in doing so beyond the simple statistical exercise. The human subject data showed substantial inter-subject variability, and the two samples also showed substantial variability. Thus, the divergence that the COGNET model showed with reference to the aggregate human responses is probably indicative of the general variability that any individual would have shown when compared to group means.

# 4. MODEL RESULTS AND EVALUATION

## 4.1 Model results

A separate report from BBN presents the data comparing model performance to human performance. Results will be summarized here. The human data used as the point of comparison is generated from 16 subjects. Eight of the subjects were tuning subjects, and these data were provided to modelers during model development as input data for model tuning. The other group of eight was fly-off subjects. As there seemed to be some differences between the subjects in the two groups, all 16 subjects' data were used for the comparison. Means and standard errors were computed for the human data for each of the six conditions (color-low WL, color-medium WL, color-high WL, text-low WL, text-medium WL, text-high WL). The primary comparisons were for overall performance for each condition, response time across all types of action with no intervening items, and subjective workload (averaged across the six subjective workload measures).

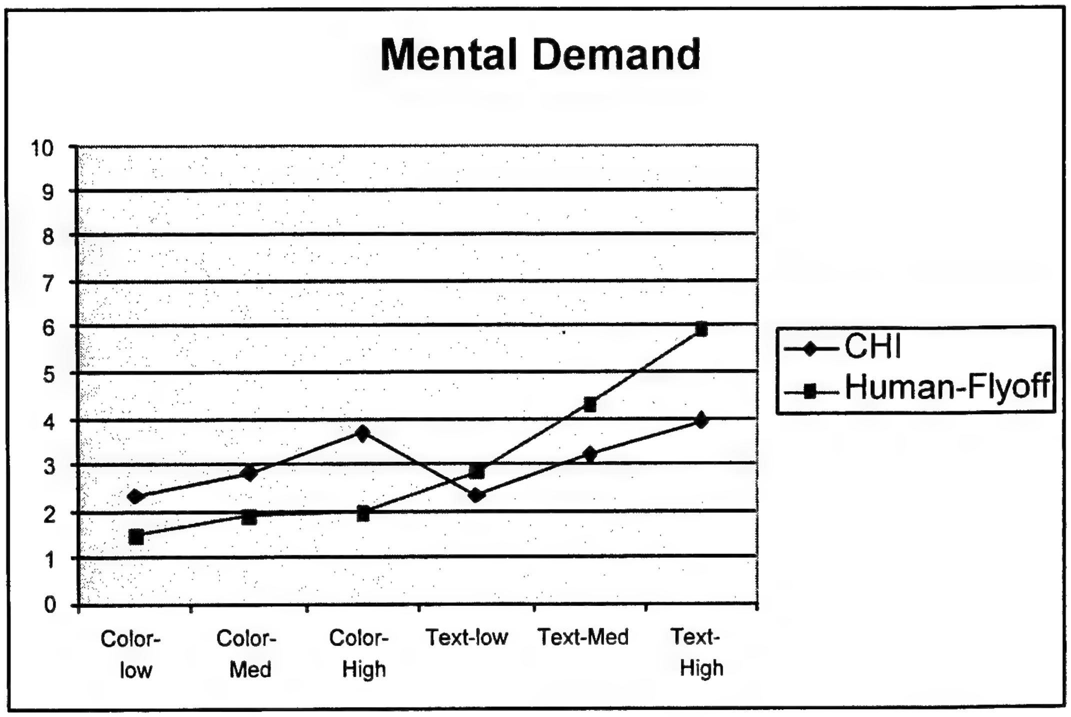


**Figure 3-9. Model versus Human Subjects on Mental Demand Measure**

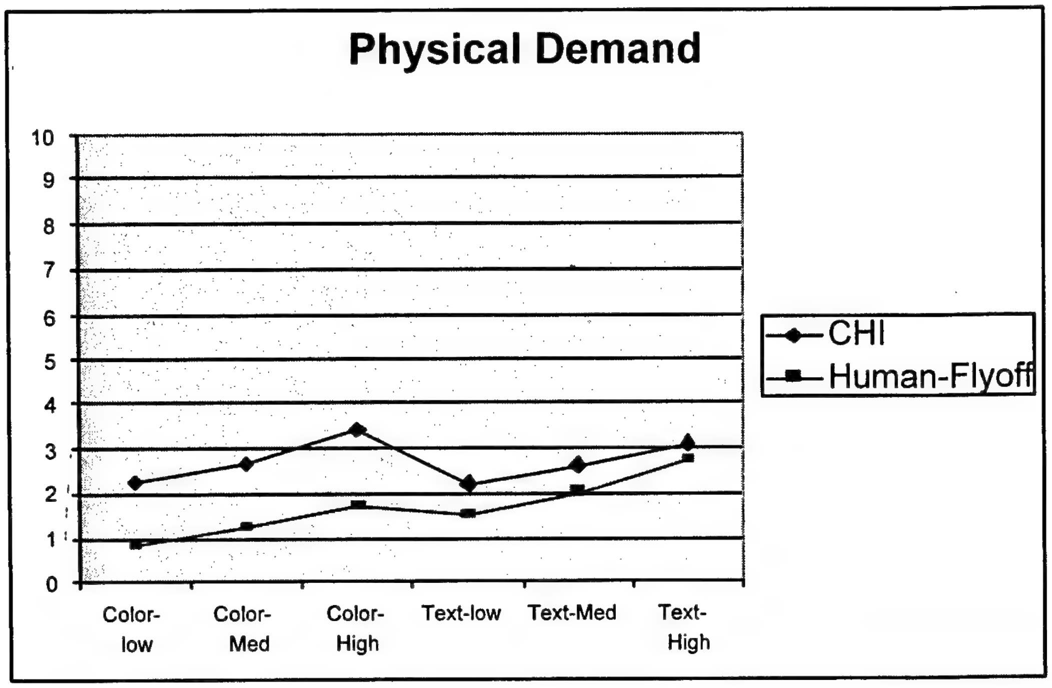


**Figure 3-10. Model versus Human Subjects on Physical Demand Measure**

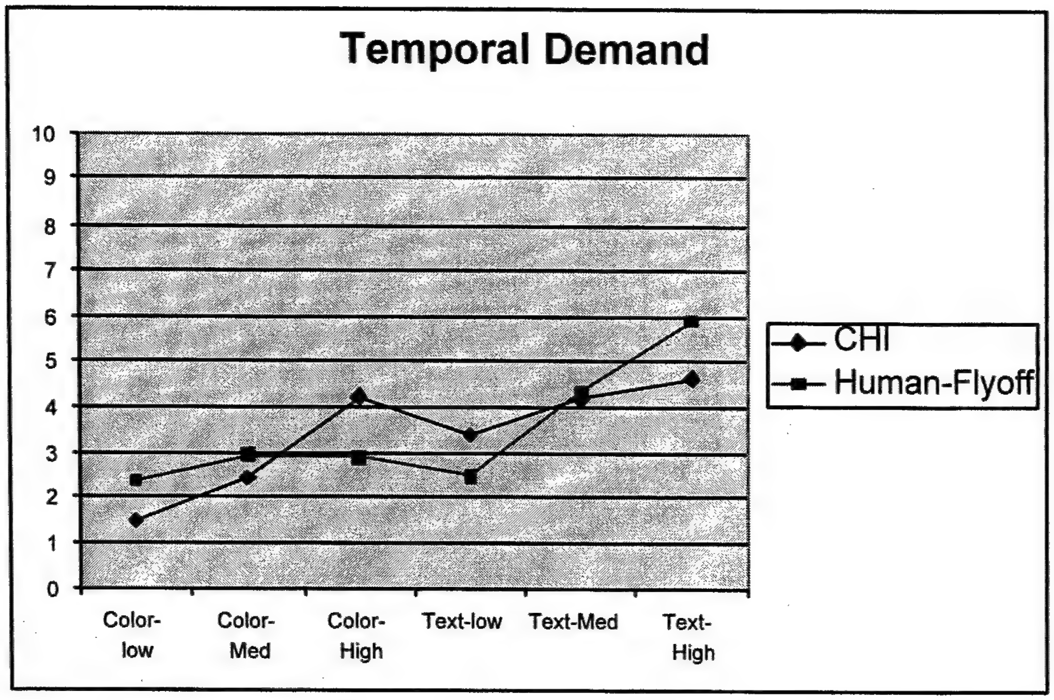


Figure 3-11. Model versus Human Subjects on Temporal Demand Measure

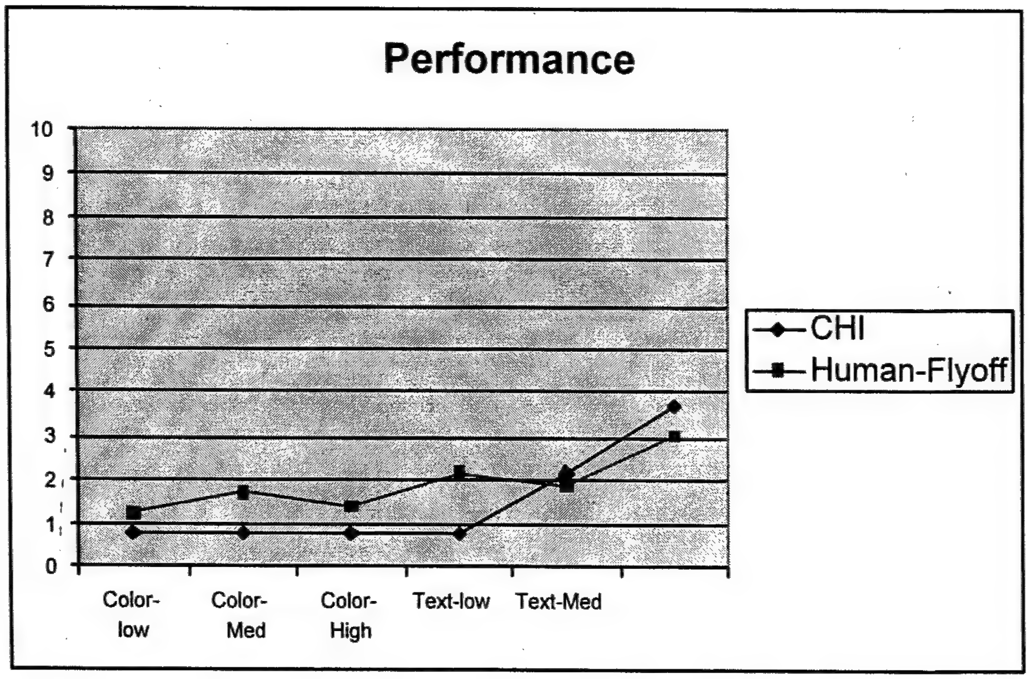


Figure 3-12. Model versus Human Subjects on Performance Measure

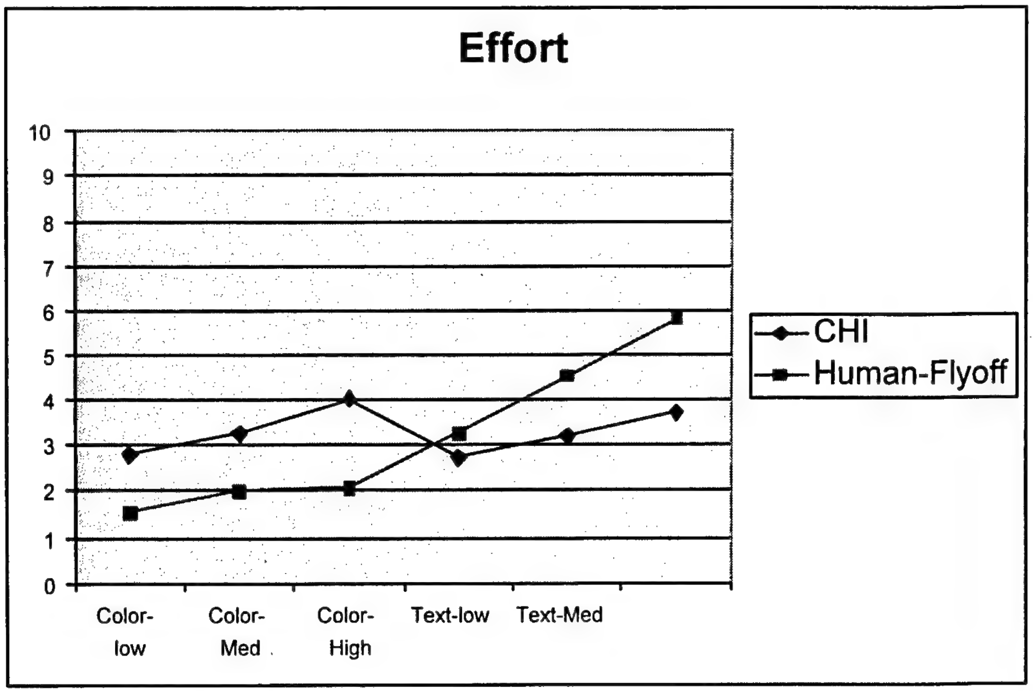


**Figure 3-13. Model versus Human Subjects on Effort Measure**

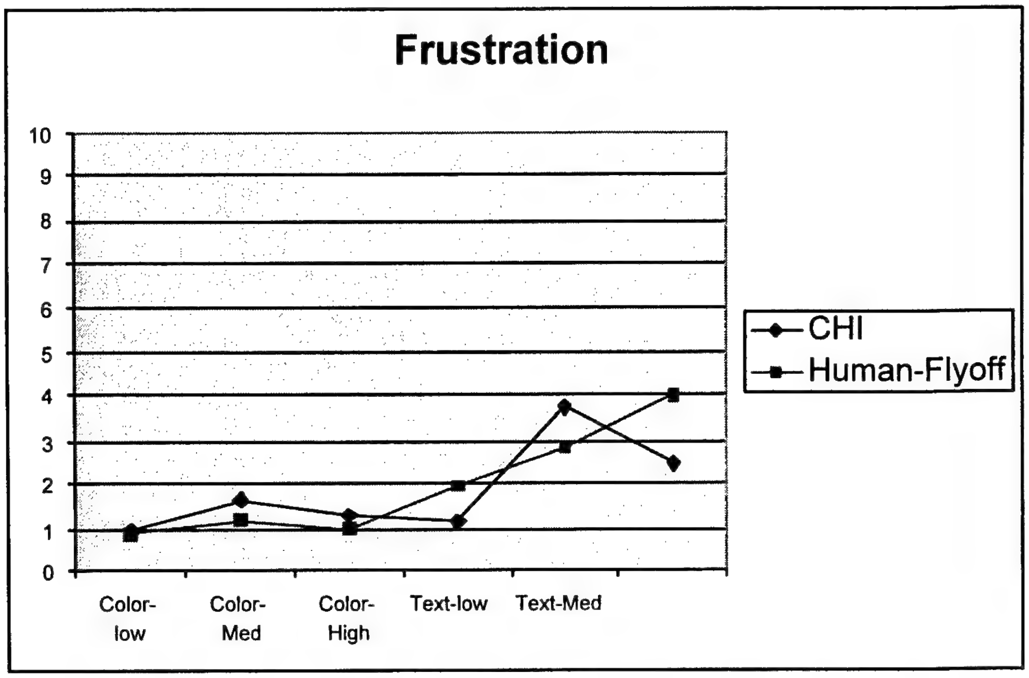


**Figure 3-14. Model versus Human Subjects on Frustration Measures**

The CHI model (of a single generic individual) made no errors on any color condition, and the number of penalty points increased in the text condition as a function of workload. These trends were very similar to the human subject data, although some human subjects made a few errors in one or more color conditions and they averaged 20 penalty points on the text-low condition, while the CHI model made no errors in that condition. Response times produced by the CHI model were comparable to the human subject data. For both the CHI model and human subjects, response times were slightly higher for all the text conditions than for all the color conditions.

Workload data has been discussed in detail in Section 3.4. Using aggregate workload ratings, the CHI model produces the same pattern of ratings as the human subjects, in that subjective workload increases with workload and the color condition is viewed as less demanding than the text condition. The CHI model tended to underestimate the difference in subjective workload between color and text conditions when compared to human subjects.

In general, the COGNET model generated results that corresponded well to human data. Although there were a few cases for which the CHI model generated a behavior that was just outside the standard error for the mean of the human subjects, the model behavior was well within the range of individual human behavior for a given condition.

## 4.2 Development process evaluation

COGNET was originally a competence-based cognitive architecture developed primarily for the purpose of supporting the development of decision aids for human-computer systems. It did not originally include any considerations of processing limitations, action or operation time, or error. Within the last two years, under another effort, it was modified into a human performance architecture, to allow better development of CGF models.

The AMBR ATC model was the first completed application using the CGF-COGNET human performance architecture. No changes to the human performance architecture were necessary to deal with the ATC-like problem. However, specific operators were added to calculate various subjective workload measures.

Two capabilities of the architecture were not exploited due to practical considerations: 1) sensory/motor modeling at a low level of granularity and 2) modeling individual differences to generate a range of performance rather than a single instance. Sensory/motor modeling at a low level of granularity could have included an eye /vision model with individual eye movement actions and accompanying micromodels, point-of-regard-based object perception, and eye movement strategies. It also could have included a hand model and elementary hand movement micromodels. It is not clear that low-level sensory/motor modeling would have improved the predictive value or generalizability of the model.

For the ATC task, we created a model of a generic individual performing the task. As it was developed, the model is not able to generate a range of performance representing the range of individual behavior expected. Such a capability could be useful in a CGF application or in a system design analysis. Stochastic error generation could be added fairly easily. Incorporating multiple strategies invoked probabilistically to represent individual difference could be done, but would require greater effort.

As mentioned in Section 3.2.7 above, our model incorporated only one error type of the many that human subjects might exhibit. Limited error modeling was a pragmatic decision that obviated the need to include various error recovery strategies in the model. The model linked error inducement to taskload above a certain level yielding a realistic error generation algorithm. Also, since this error had a ripple effect, the model was able to produce human-like performance without a large effort in this area. However, the lack of a rich model of error generation might make the model less generalizable to other, even similar, tasks.

### 4.3 Competition process evaluation

Since the impetus for the AMBR model competition was the National Research Council study (Pew & Mavor, 1998) on human performance modeling capabilities, the model competition process should be evaluated in terms of the needs addressed in that study. That is, there is a need for greater realism with respect to observable outcomes for military simulations and computer generated forces. Thus, the goals of improving realism of outcome and improving utility for military simulations should be the ultimate measures of effectiveness by which the process is judged.

One issue is whether the results of the simplified ATC task generalize to more complex military tasks. Obviously, using a real military task would have increased the cost of the program beyond the available funds and was not a practical solution. However, a middle ground of a more complex synthetic task with a greater cognitive component would be more likely to yield generalizable results and still be a manageable modeling problem.

Another area involves model assessment procedures. First, the model purpose and assessment measures (measures of effectiveness) should be defined before the model development begins, so that the models are developed with the assessment measures in mind. Second, the model comparison process should, as much as possible, stress comparison within the definition of model coverage and purpose defined. In the case of the completed competition, a new scenario and workload level meeting the criteria set forth would have been a stronger test of the models.

## 5. CONCLUSIONS

COGNET was created over the past 15 years as a method or framework for building specific models of human information processing for a range of practical purposes. It is an example of applied cognitive psychology in the tradition of Card, Moran and Newell (1983) in that the goal is to transition cognitive science to tools and techniques that can be applied to real-world human-system problems. The COGNET architecture was based on psychological theory in human cognitive processing and development of expertise, and the mechanisms incorporated built on previous cognitive modeling and artificial intelligence research.

Specifically, the COGNET cognitive architecture is based on the modified stage theory of human information processing. COGNET incorporates distinct but interdependent declarative and procedural knowledge representations. On the procedural side it supports hierarchical goal/subgoal/operation procedural knowledge structures, integration of cognitive and behavioral operations, and context-sensitive modification of strategies based on declarative knowledge content. On the declarative side, it allows the notion of mental model, or other domain-specific expert constructs to be incorporated. Attention management is an emergent property of the architecture and Principles of Operation, supporting multi-tasking easily and naturally. The more recently developed CGF variant of COGNET provides metacognition mechanisms and allows a model to be aware of its own processes and states. This cognitive proprioception allows decision-making and performance to be modified based on the state of cognitive processing. These mechanisms were used in our ATC model to drive taskload based strategies and error generation.

Subprocess theories are not an integral part of COGNET. Furthermore, no set level of modeling granularity is required, allowing adaptation to the modeling goal. Since the architecture itself is theory-neutral with respect to subprocess models, various theories could be implemented as relevant. However, the mechanisms for representing low-level processes must be built for the specific modeling application.

COGNET does not include any specific memory models. There is no explicit mechanism for decay or forgetting. Such effects can be emulated if needed for a particular application. At the present time COGNET does not have any mechanism for learning, although we are considering methods for incorporating learning in the near future.

iGEN™ is a COTS product for developing and running COGNET expertise models. It includes graphical authoring tools and debugging tools, as well as a facility for compiling standalone models that can be embedded within a larger software system. As a commercial system, there are accompanying courses, manuals, and support. In addition to a stable commercial version, various versions of iGEN™ can coexist to support research extensions for accommodating new modeling demands, such as metacognition or micromodels for human performance modeling. New features that are of general interest can then be incorporated in the COTS product.

## 6. REFERENCES

Broadbent, D. (1958). Perception and Communications. New York: Pergamon Press.

Card, S., Moran, T., & Newell, A. (1983). The Psychology of Human Computer Interaction. Hilsdale, NJ: Lawrence Erlbaum Associates.

Carver, N., & Lesser, V. (1994). Evolution of blackboard control architectures. Expert Systems with Applications, 7(1), 1-30.

Chi, M., Glaser, R., & Farr, M. (1988). The nature of expertise (Hillsdale, NJ: Lawrence Erlbaum Associates).

Englemore, J., & Morgan, T. (1988). Blackboard systems. Reading, MA: Addison-Wesley.

Ericsson, K., & Kinsch, W. (1995). Long-term working memory. Psychological Review, 102(2), 211-245.

Ericcson, K., & Smith, J. (Ed.) (1991). Toward a general theory of expertise: Prospects and limits (Cambridge: Cambridge University Press).

Erman, D., Hayes-Roth, F., Lesser, V., & Reddy, D. (1980). The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. ACM Computing Survey, 12, 213-253.

Fitts, P. (1954). The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47, 381-391.

Hart, S. & Staveland, L. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In Hancock, P. & Meshkati, N., Human Mental Workload. Amsterdam: North-Holland, 139-183.

Hayes-Roth, B. (1985). A blackboard architecture for control. Artificial Intelligence, 26, 251-321.

Gray, W., & Boehm-Davis, D. (in press). Milliseconds Matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. Journal of Experimental Psychology: Applied.

Hayes-Roth, B., & Hayes-Roth, F. (1979). A Cognitive Model of Planning. Cognitive Science, 3, 275-310.

Hoffman R. (1992). The Psychology of Expertise: Cognitive Research and Empirical AI (New York: Springer-Verlag).

Hofstadter, D. (1979). Godel, Escher & Bach: An Eternal Golden Braid. Basic Books, New York.

Kieras, D., & Meyer, D. (1995). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. (EPIC Tech. Rep. No. 5 (TR-95/ONR-EPIC-5). Ann Arbor, MI: University of Michigan, Electrical Engineering and Computer Science Department.

Klein, G. (1989), Recognition-primed decisions. In W.B. Rouse (Ed.), Advances in Man-Machine Systems Research, Vol. 5 (Greenwich, CT: JAI Press), 47-92.

Klein, G., Orasanu, J., Calderwood, R., & Zsambok, C. (Eds.) (1993). Decision making in action: Models and methods. Norwood, NJ: Ablex.

Kolodner, J. (1988). Proceedings of the Case-Based Reasoning Workshop. San Mateo, CA: Morgan Kaufmann.

Kolodner, J. (1993). Case-Based Reasoning. San Mateo, CA: Morgan Kaufmann

Lane, N., Strieb, M., Glenn, F., & Wherry, R. (1981). The Human Operator Simulator: An Overview. In J. Moraal and K. F. Kraiss (eds.), Manned Systems Design: Methods, Equipment, and Applications. New York: Plenum Press.

McGaugh, J. (2000). Memory—a century of consolidation. Science, 287, 248-251.

Meyer, D., & Kieras, D. (1996). A Computational Theory of Executive Cognitive Processes and Human Multiple-Task Performance: Part 1. Basic Mechanisms [EPIC Report No. 6

(TR-96/ONR-EPIC-06)]. Ann Arbor, MI: University of Michigan (to appear in Psychological Review, 1997).

Newell, A. (1990). Unified theories of cognition. Cambridge, MA: Harvard Univ. Press.

Nii, P. (1986a). Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures PART ONE, AI Magazine, 7,(2), 38-53.

Nii, P. (1986b). Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures PART TWO, AI Magazine, 7(3), 82-106.

Pew, W. and Mavor, A. (1998). Modeling Human and Organizational Behavior: Applications to Military Simulations. (National Research Council) (Chapter 3, pp. 51-111). Washington, D.C.: National Academy Press. Also available at http://www.nap.edu.

Pylyshyn, Z. (1985). Computation and cognition: Toward a foundation for cognitive science. Cambridge MA: MIT Press (Bradford Books).

Pylyshyn, Z. (1989). Computing in cognitive science. In M.I. Posner (Ed.), Foundations of cognitive science. Cambridge, MA: MIT Press.

Reiger, C. (1977). Spontaneous computation in cognitive models. Cognitive Science, 1(3).

Selfridge, O. (1959). Pandemonium: A paradigm for learning, in Proceedings of the Symposium on the Mechanization of Thought Processes, pp. 511-529.

Turing, M. (1950). Computing machinery and intelligence. In Mind: reprinted in A.R. Anderson (Ed.), Minds and Machines (1964) Englewood Cliffs, NJ: Prentice-Hall.

VanLehn, K. (1996). Cognitive skill acquisition. Annual Review of Psychology, 47, 513-539.

Weiland, M., and Eilbert, J. (2000) ORGNET/PRO: Methodology and Program for Redesigning Organizations. (CHI Systems Technical Report No. 000320.9905). Lower Gwynedd, PA: CHI Systems, Inc.

Weiland, M., Zachary, W., and LeMentec, J. (1998) ORGNET/PRO Methodology and Toolset for Redesigning Organizations. Proceedings of the Human Factor and Ergonomics Society 42nd Annual Meeting, Human Factors and Ergonomics Society: Chicago, IL.

Zachary, W., Le Mentec, J., & Glenn, F. (2000) Developing computational models of metacognition in support of manning reduction technology development (CHI Systems Technical Report No. 000115). Lower Gwynedd, PA: CHI Systems, Inc.

Zachary, W., Le Mentec, J., & Ryder, J. (1996). Interface agents in complex systems. In Ntuen, C. and Park, E. (Eds.), Human interaction with complex systems: Conceptual Principles and Design Practice. Norwell, MA: Kluwer Academic Publishers.

Zachary, W. & Ryder, J. (1997). Decision support systems: Integrating decision aiding and decision training. In Helander, M., Landauer, T., & Prabhu, P. (Eds.) Handbook of Human-Computer Interaction, 2nd Edition. Amsterdam, The Netherlands: Elsevier Science.

Zachary, W., Weiland, M., and Le Mentec, J. (1997) Allocating Cognitive and Decision-Making Functions From Cognitive Models of Team Requirements, in Proceedings of the First International Conference on Revisiting the Allocation of Functions Issue. Galway, Ireland. National University Ireland.